REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	7 August 1998	

4. TITLE AND SUBTITLE
A CAK METHODOLOGY FOR KNOWLEDGE ASSISTED DESIGN

5. FUNDING NUMBERS

6. AUTHOR(S)
Paul A. Hey

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Purdue University

8. PERFORMING ORGANIZATION REPORT NUMBER
98-050

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
THE DEPARTMENT OF THE AIR FORCE
AFIT/CIA, BLDG 125
2950 P STREET
WPAFB OH 45433

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION AVAILABILITY STATEMENT
Unlimited distribution
In Accordance With 35-205/AFIT Sup 1

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

14. SUBJECT TERMS

15. NUMBER OF PAGES
91

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT

A CAD METHODOLOGY FOR KNOWLEDGE ASSISTED DESIGN

A Thesis
Submitted to the Faculty

of

Purdue University

by

Paul A. Hey

In Partial Fulfillment of the
Requirements for the Degree

of

Master of Science

December 1997

To Ruth

ACKNOWLEDGEMENTS

The successful completion of this work would not have been possible without the guidance, assistance, and support of many people and organizations. Although the mere mention of their name does not fully express the gratitude I hold for them, I would like to express my appreciation for those who helped me survive this experience.

First and foremost I must thank my advisor Professor David C. Anderson for his advice, guidance, and friendship throughout this process. His dedication to learning and development of a first-class research facility have made this an experience I will always cherish and never forget. I would also like to thank the other members of my advisory committee, Professors T. C. Chang and K. Ramani. I must also extend my thanks to Professor Warren Waggenspack of Louisiana State University, a former CADLABer who introduced me to computer-aided design, Purdue, and Professor Anderson.

I gratefully acknowledge the United States Air Force Institute of Technology (AFIT) for accepting me as a Second Lieutenant fresh out of college to pursue my Masters Degree in Mechanical Engineering. In particular, I thank my program managers Captain Richard Baker and Lieutenant Scott Naylor, and the Purdue AFROTC staff for helping to keep my Air Force life in order.

Many thanks to the staff of the Center for Collaborative Manufacturing, Dr. Nancy Bulger for giving me a temporary home, and Pat Smith, Leah Peffley, and Barbra Thayer for letting me fix their computers in exchange for mints, sandwiches, and great conversation. The support I received under grant EEC-9402533 to the Center for Collaborative Manufacturing made it possible for me to be accepted into AFIT's Scholarship, Fellowship, and Grant Program. I am also thankful to the staff of the

Mechanical Engineering Graduate School Office, Carol Wolfe, Chris Kitterman, and Paul Neulieb for keeping my academics issues in order and knowing how to fill out those pesky registration forms.

I would also like to thank Cummins Engine Company, for their contributions to this research which allowed us to apply our ideas to a real-world problem. Special thanks go to Joe Mickel for helping establish the collaboration and Joy Lindsay, Mike Marthaler, Dave Richter, and Jim Fleming for teaching me about the flywheel and answering my many questions.

It has been a pleasure working with the other students and staff in the CADLAB, David's B. G. and H., Roger, Joe, Percy, Mark, Mike, Charles and Rajaraman. Without their help and friendship, I could not have complete this work or learned how to excel at so many computer games. I would also like to thank my fellow AFIT students, Captain Monty Greer, Captain Drew Causey and Major Mark Knoff who are friends and, along with the CADLAB students, managed to survive my talent for twisting ankles on the basketball court.

Finally, I am grateful to my family whose support and guidance kept me going. My parents, George and Linda Hey, and my wife's parents, Norman and Elaine Davidson, provided moral support along the way. My dogs, Keeper and Vagrant, deserve thanks for getting me to occasionally walk in the park after the long days of deep thought and programming. And, most importantly, I would like to thank my wife, Ruth. I am forever grateful for her continuous love and support, especially as the hours grew longer and our time together shorter. Without her, I could not imagine where I would be.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Hey, Paul Andrew. M.S.M.E., Purdue University, December 1997. A CAD Methodology for Knowledge Assisted Design. Major Professor: David C. Anderson, School of Mechanical Engineering.

Modern computer-aided design (CAD) systems have developed into integral support tools for the product design and development process. Designers must, however, draw upon experiential engineering knowledge such as past experiences, specific design rules and procedures, and heuristic reasoning just as before the advent of CAD. This work develops a methodology for integrating experiential engineering knowledge in an interactive CAD environment that serves as a knowledgeable design assistant and supports a design process controlled by the designer.

The knowledge assisted design environment is an object-oriented, domain independent framework based on a blackboard architecture that incorporates a feature-based design environment with multiple, autonomous knowledge sources. The system can be utilized for any domain for which a set of features and knowledge sources have been defined. The knowledge sources provide design assistance by reacting opportunistically to a developing design solution and by presenting advice interactively to the designer. The object-oriented knowledge and hierarchical, feature-based model representations are presented along with the design environment and its functionality. The methodology is applied to the industrial application of engine flywheel design.

CHAPTER 1

INTRODUCTION

Design is a cognitive process that requires a designer to apply both past experience and general engineering knowledge to achieve a completed product. Since the advent of computer-aided design (CAD) in the early 1960's, the computer has become an integral part of the product design and development process. Early CAD systems provided the capability to create two-dimensional engineering drawings to document a completed design. Using modern CAD systems, however, a designer can a construct a sophisticated three-dimensional solid model of the design artifact and perform complex engineering analyses on that model. These advances have significantly lessened the costs incurred during the design process by reducing the number of design cycle iterations to arrive at a completed design.

Despite the utility of today's conventional CAD systems, their capabilities fall short of the original vision of the computer's role in the design process. Regarded as one of the first CAD systems, Sutherland's Sketchpad [SUTH63] sparked discussion of how the computer would support designers in the future. The systems were envisioned to act as intelligent design assistants and interactive support tools that would dynamically assist the designer throughout the design process. Conventional CAD systems, however, have only partially fulfilled that goal. The systems serve as sophisticated replacements for the slide-rules and drafting tables of the past and, because they are designed to support a wide range of engineering domains, inherently restrict the amount of design assistance they can provide. Rather than actively assisting the designer by playing an integral role in the design

process, modern CAD systems have developed into geometric modeling and analysis tools that only partially support the design process. As shown in Figure 1-1, the designer must draw upon experiential engineering knowledge such as past experiences, specific design rules and procedures, and heuristic reasoning just as before the advent of CAD.

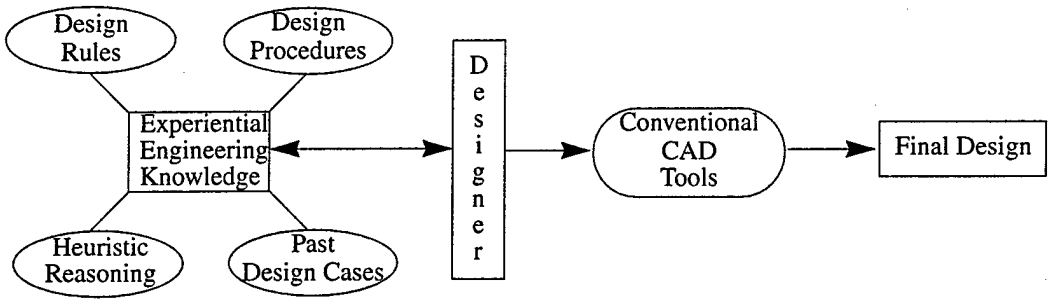


Figure 1-1. Conventional CAD Methodology

1.1 Integrating Engineering Knowledge

To move CAD closer to its original intent of serving as both support tool and intelligent design assistant, the design systems must be refined to include representations of low level engineering information, such as points and lines, and high level engineering information, such as material properties and functional properties, from which they can reason to provide the envisioned design assistance. Incorporating engineering knowledge into an interactive design environment provides the basis for developing a CAD system that serves as a knowledgeable design assistant. The environment will also serve as an engineering knowledge repository by maintaining important design rules and heuristics long after the designer responsible for contributing the design knowledge has left the organization.

The requirements for the successful integration of engineering knowledge fall under three main categories: the product model, the design knowledge, and the design

environment. First, the model of the design artifact must contain high level engineering content that allows the designer to express the engineering characteristics of the design beyond geometry alone. The designer must be able to accurately express the engineering attributes of the design model using terminology familiar to, and appropriate for, the design domain in which the designer is working. At the end of the design process, the model must also exist as a complete, stand-alone entity that is independent of the process that created it.

Second, the representation of the design knowledge must be developed to interact with the model to provide active assistance during the design process. The knowledge must also express both the geometric and non-geometric engineering attributes of the design artifact and the design process in a terminology familiar to the designer. The knowledge representation must be adaptable to the varied knowledge utilized during the design process, and it must be developed to allow a designer to easily add new engineering knowledge to the pool of domain knowledge without the involvement of an outside source and a detailed understanding of the underlying design system functionality.

Finally, a design environment must be developed in which the designer and the engineering knowledge can interact cooperatively to arrive at a satisfactory design. Most previous attempts at integrating experiential engineering knowledge and conventional CAD methodologies have resulted in automated, knowledge-based systems developed for specific domains that required the designer to provide only the initial design specifications for the design process to proceed. While automated design systems have proven to be successful in design domains where there is very little creativity and variety in the design process, the majority of engineering design fall outside that category. Attempts at automating a design process that involves any human cognitive capabilities have proven to have only limited applicability. Therefore, the design environment must be developed such

that the designer is in complete control of the design process and the engineering knowledge provides interactive design assistance during the design process.

1.2 Research Objectives

The objective of this research is to develop a computer-aided design methodology for knowledge assisted design, as shown in Figure 1-2, that integrates engineering knowledge in an interactive design environment that serves as an interactive design assistant and dynamically supports a design process controlled by the designer. The goal is to develop a domain-independent framework that can be applied to varied domains by adding domain specific engineering knowledge to the design environment, as opposed to a general design system that can be used for any domain of engineering design.

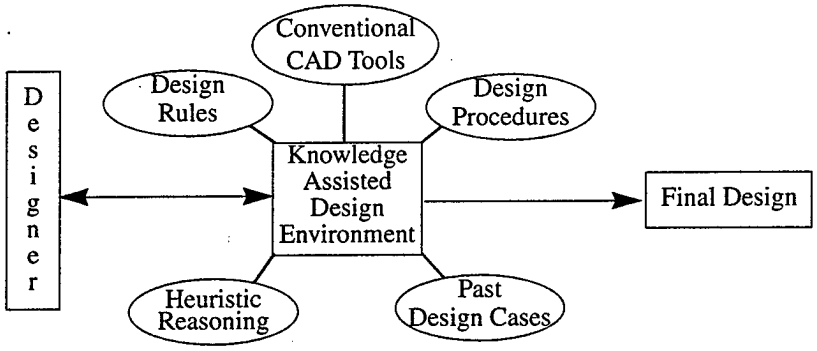


Figure 1-2. Knowledge Assisted Design Methodology

This primary goal can be divided into three tasks. First, a product model must be developed that allows the designer to adequately represent both the geometric and non-geometric attributes of an evolving design solution. The model structure must be easily extensible and sufficiently general to support as broad a range of domains as possible, and it must support interaction with the engineering knowledge to provide design assistance that actively supports the design process. Second, a knowledge representation must be

developed that can also be easily applied to varied design domains and accurately represent the wide range of engineering knowledge utilized during the design process. Finally, a domain independent design system framework must be developed that incorporates the designer, the model, and the knowledge in an environment that allows the designer to dictate the process by which a product model is constructed while receiving dynamic, interactive design assistance from the engineering knowledge.

1.3 Thesis Overview

Chapter Two presents a discussion of past research efforts in the knowledge and model representation techniques and the integration of engineering knowledge and conventional CAD methodologies. The model and knowledge representations implemented for this research are discussed in Chapter Three. Chapter Four presents the framework for integrating these representations for knowledge assisted design. Chapter Five presents a prototype knowledge assisted design system applied to the specific domain of engine flywheel design. This system was developed for an actual industrial application to provide design assistance using Society of Automotive Engineers (SAE), International Organization for Standardization (ISO), and corporate design standards. Finally, Chapter Six provides concluding remarks and directions for future research.

CHAPTER 2

BACKGROUND AND PAST RESEARCH

An examination of the past research efforts in related fields provides the background necessary to better understand the pertinent issues of developing a methodology for knowledge assisted design. The investigation of developing intelligent CAD systems for design assistance has been ongoing since computers were introduced into the realm of engineering design. Much of the research was directed towards the integration of engineering knowledge into CAD systems and the appropriate means to represent the engineering knowledge for the design environments. This chapter reflects on those previous efforts and identifies the key research issues for the development of a knowledge assisted design methodology.

2.1 Artificial Intelligence in Design

The first attempts at developing CAD systems that drew upon experiential engineering knowledge occurred during the growth of Artificial Intelligence (AI) in the late 1970's. As computers became more prevalent and powerful, a concerted research effort investigated the development of computer programs that behaved and performed actions like humans. AI offered CAD developers the tools to reason about engineering information in manners previously unavailable. The AI systems operated using sophisticated symbolic reasoning techniques and were developed in symbolic processing languages such as LISP and Prolog. These systems exhibited limited success and are still in use today primarily in

automated design and diagnostic systems. Despite their inherent trend towards design automation, the AI methodologies highlight some of the key aspects and provide some of the tools necessary for developing a useful design assistant system.

2.1.1 Knowledge-Based Expert Systems

The first AI systems used in design were knowledge-based expert systems (KBES), which attempt to integrate heuristic reasoning and general domain expertise into computer programs that behave as experts in their respective domain [AKMA94]. Researchers hypothesized that human experts reason from a set of learned rules and guidelines to arrive at a final design. KBES attempted to mimic this human reasoning process and offered automated design support.

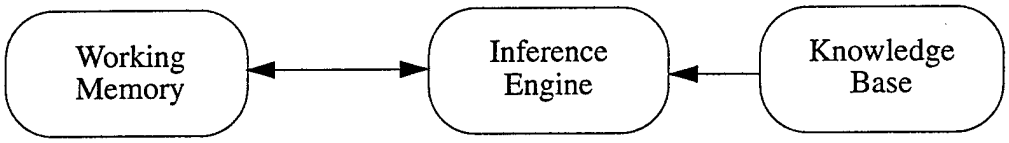


Figure 2-1. The knowledge-based expert system structure

The basic structure of the KBES, as shown in Figure 2-1, consists of three components: the knowledge base, the inference engine, and the working memory. The design process is controlled by the application of design rules and processes stored in the knowledge base. The inference engine contains the reasoning mechanisms by which the knowledge base is searched and specific rules are applied to answer queries concerning the design solution as it develops in the working memory. The distinctly separate inference engine and knowledge base allow for domain independent KBES shells that only required a new knowledge base to perform design in a particular domain [DIXO95].

KBES research and development concentrated in two areas: knowledge acquisition and inference engine reasoning algorithms. Knowledge acquisition, the gathering and coding of domain-specific rules, evolved into a very sophisticated process. Specialized knowledge engineers interviewed experts in the specific design domain, and attempted to acquire all of the appropriate rules and actions required to achieve a particular design. As the KBES moved from the beginning stages of addressing example design problems to attacking actual design problems, however, the size and complexity of the knowledge bases made the approach impractical. New inference engine reasoning algorithms were developed to accommodate the growth of the knowledge bases, and research focused on generating more efficient, sophisticated knowledge space search algorithms, conflict mitigation techniques, truth maintenance systems, and decision explanation facilities.

The KBES represent the thought processes and cognitive abilities of humans as a collection of possible alternative decisions coded into the knowledge base. The results are design systems programmed to solve a very complex problem yet have difficulties solving simple problems within the same domain. In an attempt to represent fully the knowledge necessary to complete any fairly complex design task, the knowledge base grew into large, intricate decision trees that were difficult to maintain. The KBES also demonstrated brittle functionality. The solution process would fail if, during the design process, the input data was not specified in precisely the correct manner or a design solution developed that was not represented in the knowledge base. The addition or modification of any of the design rules required the entire knowledge base to be reprogrammed.

The results of these research efforts led to several KBES for design. The PRIDE project presented in [MITT86] automated the design process for paper-handling systems in photocopiers, and [PACK94] implemented a KBES for the mold transport substructures at Sikorsky aircraft. The AIR-CYL system implemented an expert system for the design of air cylinders [BROW92]. Despite the limited success of these and other KBES, they have

not proven to be adequate for providing active assistance during the design process. As [DUFF96a] argues, the process of integrating human expertise and CAD systems may lend itself to some form of automation, yet removing the designer from the design process eliminates the substantial benefits of a cooperative design effort between the designer and the computer and is also restrictive.

2.1.2 Model-Based Reasoning

A new reasoning methodology known as model-based reasoning emerged to overcome the inadequacies of the KBES. The model-based reasoning methodology is based on a different view of the human cognitive process during design then the KBES. In design, not all of the engineering information can be expressed in terms of if-then rules. Engineering design is based on physical models, mathematical formulae, fundamental engineering principles and imprecise subjective concepts. Model-based reasoning contended that searching the design space for the possible set of existing elemental representations to form the completed design was a more accurate representation of the human design process than reasoning by searching the possible set of design decisions as hypothesized for the KBES [FALT96].

The system structure for the model-based systems is very similar to that of the KBES. Instead of reasoning on a single set of rules, however, the inference engine searches a set of representations of the design artifact's components and their interactions based on mathematical and physical principles that described their function [GERO88]. In response to initial input specifications from the designer, the system will iterate on the design parameters based on the current state of the design solution until an adequate design is achieved.

Much of the research for applying model-based reasoning to design advanced the development of sophisticated languages to represent accurately the design artifact's

components and their interactions [JOSK96], [ROSE94]. The model-based reasoning approach was applied to a variety of design automated applications, from architectural design code verification [DYM88], to the design of small mechanical assemblies [GOEL89]. The intelligent boiler design system presented in [RIIT88] automated the boiler plant design process using the model-based reasoning methodology. The components of a boiler plant such as the pumps, pipes, circuit breakers, and conductors were represented within the design system. After the designer provided the initial functional specifications of the boiler plant, the design system arrived at a final plant design based on the component properties and interactions defined in the various component representations.

The model-based reasoning methodology was successfully applied to the medical and system diagnosis fields [ABU94], [GOEL96]. Functional systems, such as the human vascular system or a chemical treatment facility, were represented in model-based diagnostic environments from which accurate diagnoses can be made. The actual system functionality was compared to the representation, and any deviations were presented to the user. The GUARDIAN system presented in [HAYE92], for example, supervised hospital intensive-care units and notified the supervising medical personnel if the units failed to perform their expected tasks. This model representation, however, did not fit well into the design assistance field because modeling the design process is not equivalent to modeling a functioning system. Design is an open-ended process that varies from one designer to another and can only be fully defined for a few design domains.

The model-based systems provided an impetus for later AI-based CAD systems to shift from knowledge-based systems that based their functionality on the rules that controlled the design process to object-based systems that utilize domain knowledge to enhance the functionality of a design system based on the development and analysis of a product model. One of the main shortcomings of the KBES was their attempt to fully

define and control the design process using only knowledge about the domain without a clear representation of the design artifact. The model-based systems attempted to alleviate this problem by representing both the knowledge about the model and model itself in one knowledge resource. This trend progressed with the continued integration of AI and CAD.

2.1.3 Case-Based Reasoning

Case-based reasoning is an extension of the model-based reasoning methodology. Rather than providing representation based on fundamental engineering principles or rules on which decisions and reasoning can be based, case-based systems use actual design cases as the knowledge base. The case-based reasoning methodology is founded on the principle of design by analogy - designers develop new design solutions based on their past experiences in that domain [KUMA95].

The case-based framework is shown in Figure 2-2. When presented with a new design problem, the system retrieves similar past designs from the case base by matching design attributes present in both the current design specifications and the past design cases. It then attempts to adapt and modify the current problem data using the past solution strategies to arrive at a complete design artifact. The new solution is then added to the case base for use in later design problems.

This methodology exhibits a distinct shift towards object-based systems. The adaptation and modification of the design model is based on comparisons between the current design specifications and the past design cases. The solution is achieved by an interaction between the model and the knowledge about the model, not just the knowledge itself. There is also a clear separation between the design artifact and the knowledge resource.

Researchers developed a number of design systems that utilized the case-based reasoning methodology. The DEJAVU system performed functional mechanical gear

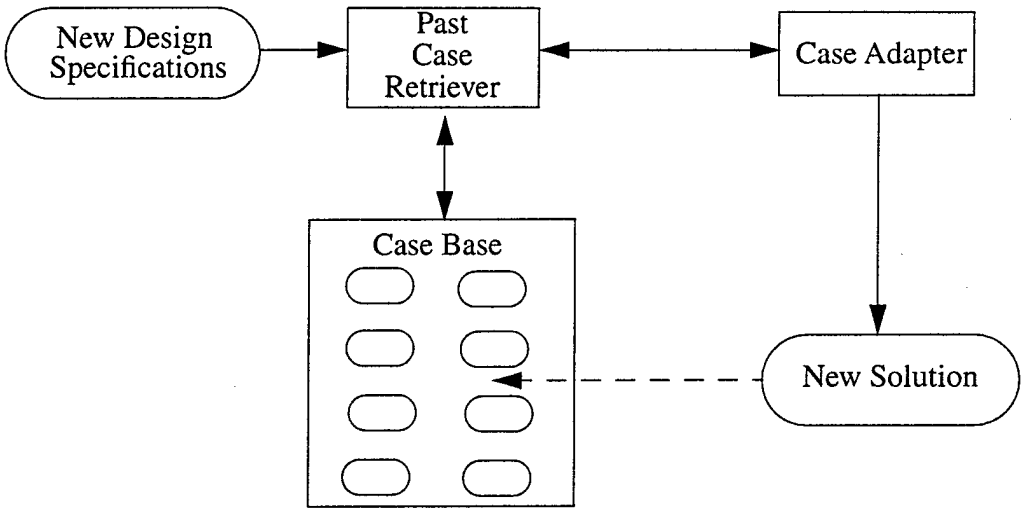


Figure 2-2. Case-Based Design Framework

design by retrieving past gear design cases and adapting them to solve the current design problem based on functional design specifications [BARD93]. CLAVIER provided manufacturability assessments of aircraft engine parts [HINK95], and the PANDA system addressed the need to simplify the pumper engine design process for novice designers [RODE93]. These systems applied case-based reasoning to perform an automated design process based on start-up data supplied by the designer. [SYCA92] describes the CADET system for design synthesis of fluid-mechanical devices. The past design cases are represented as models that describe the physical form and qualitative function of various devices. When the functional requirements for a new design are provided by the designer, the system retrieves relevant past design cases that can contribute to the new design solution. The form and function of the new design is synthesized based on qualitative reasoning guided by design rules embedded in the case adapter. Several other case-based systems for design can be found in [MAHE95].

Case-base researchers forwarded the concept of "knowledge chunking" as the means for developing an efficient method for searching the large case bases of the prototype design systems. Dealing with a case as a whole was neither effective nor efficient. Instead, a case was decomposed into smaller components, or chunks, of knowledge. [DOME93] contends that by decomposing a past design cases into smaller, more manageable components of the design artifact, the reasoning process becomes more efficient and effective by only searching for the relevant parts of past design cases, not the case as a whole. This provided a more effective means of organizing the past design cases and forced the system designers to adequately decompose the design artifact into the applicable and pertinent components of the domain. Retrieval of past design cases, however, mandated new design problems match the decomposed structure of past cases. This required prior knowledge of past design cases and limited the case-based approach to addressing design problems for rigidly defined domains.

2.1.4 Blackboard Architecture

The blackboard architecture is another problem solving methodology developed by AI researchers that has been applied to CAD systems. The blackboard problem solving methodology is based on the concept of an opportunistic problem solving environment, which is analogous to a group of experts gathered in a room attempting to solve a problem. The experts are unable to discuss the problem with one another and are only permitted to communicate using a blackboard at the front of the room. A moderator mediates the actions of the experts and controls which expert goes to the board to contribute to the developing design solution. Each expert contributes to the solution by reacting opportunistically to changes in the solution and notifying the moderator of their desired action. As the solution develops on the board, more experts are able to contribute until the problem is solved.

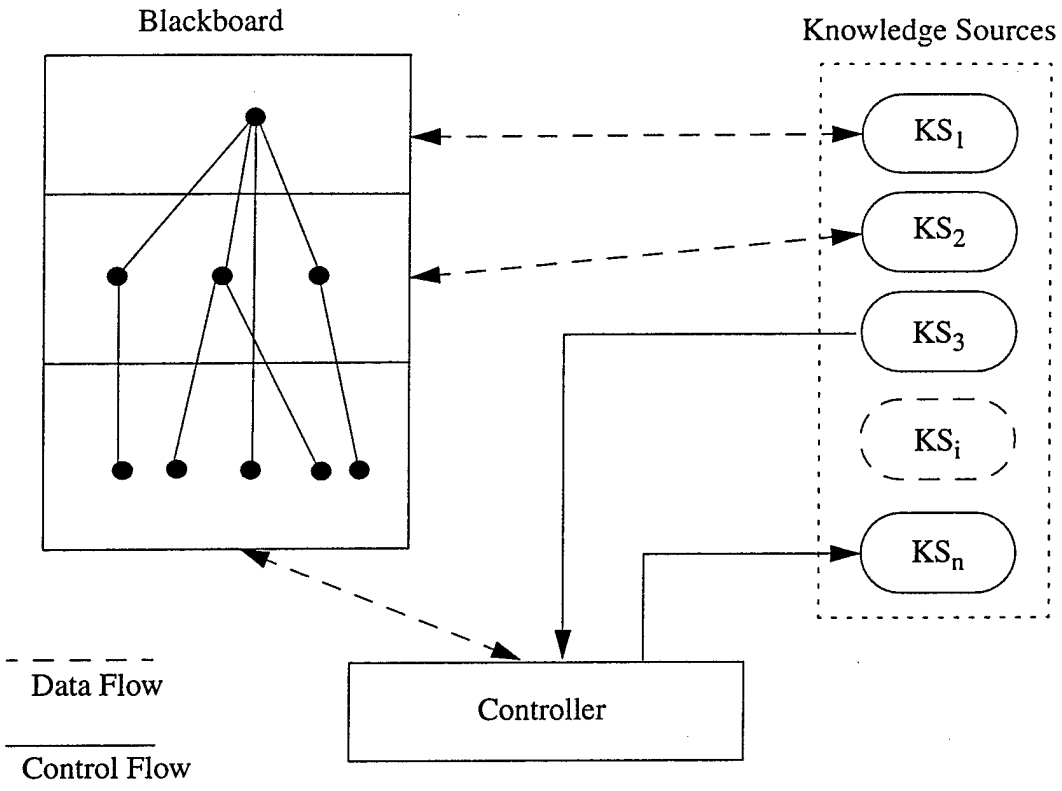


Figure 2-3. The blackboard system framework

The actual blackboard system structure, shown in Figure 2-3, consists of three primary components: the blackboard, the controller, and the knowledge sources. The blackboard is the global solution space that contains the developing solution. During the development of the solution, all interaction is accomplished via the blackboard and the knowledge sources do not directly interact. Those knowledge sources that can apply themselves to the design solution notify the controller which moderates the order in which the knowledge sources apply themselves to the solution. As the solution proceeds, the knowledge sources react opportunistically to it, manipulating the objects represented on the blackboard, until arriving at an acceptable solution.

This framework differs significantly from the previously discussed systems because more than one knowledge resource contributes to the solution. The knowledge sources are

maintained separately from one another and encapsulate a single aspect of the domain knowledge. This organization eliminates the problem of the large, deeply nested knowledge bases found in the KBES or a single, complex model used in model-based reasoning systems. It also allows for new knowledge sources to be brought into the system without reorganizing the entire knowledge base. Separate knowledge sources allow various types of knowledge to be used in support of the solution process and provide a more realistic knowledge representation than the previous methodologies. Unlike KBES and model-based systems, the order in which the blackboard knowledge sources apply themselves to the solution is independent of their grouping within the system.

The blackboard framework also incorporates the concept of an object-based system. The blackboard framework clearly separates the model from the knowledge and utilizes both the knowledge about the domain and the actual model of the design artifact to complete the design. The blackboard can be divided into separate sections that represent a different aspect of the solution domain. This decomposes the model along similar lines as case-based systems. The system also provides an interactive link between the model and the various knowledge sources.

Blackboard systems were originally developed for use in speech recognition, system diagnosis, and instructional planning environments. [ENGE88] and [JAGA89] provide a review of these early blackboard systems. After experiencing early success, researchers turned to blackboard frameworks for developing knowledge-based CAD systems [BUSH87], [CORB86], [DIXO84], [SRIR86], [VENK86]. Most of the systems utilized the blackboard framework's multiple knowledge sources and opportunistic problem solving methodology to facilitate automated design. [MAYE88] reports a blackboard system for mechanical design that integrates the designer in the design process. Multiple knowledge sources cooperate to select appropriate design components based on the input specifications supplied by the designer and acquired from the designer during an interview

process similar to those used in KBES. The designer actively participates in the design process by acting as a knowledge source and defining new design rules as the design solution develops.

2.1.5 Hybrid Systems

Hybrid systems developed from the realization that limiting the knowledge resources to one type of knowledge representation did not adequately support the design process. A hybrid system combines two or more of the previously discussed AI methodologies in a single environment. [CHAM95] discusses the importance of creating hybrid systems to integrate multiple knowledge representations in the same design environment and to create a more robust design system. Each AI knowledge representation has its strengths and weaknesses, yet no single type completely solves the design problem.

In addition to combining AI tools, the hybrid systems also combined AI methodologies with conventional CAD tools, such as analysis and drafting packages. [RODR94] presents a hybrid design system developed for shape definition of structural components using a combination of an expert system and an analysis program. The integration of the various systems into one environment does not mean, however, they will work cooperatively towards a viable solution. These design environments have developed into integrated systems that utilize the various AI techniques for solving automated design problems in a procedural manner, stepping through a series of pre-defined design stages. These environments do not provide the necessary framework for supporting design assistance.

2.1.6 Commercial Packages

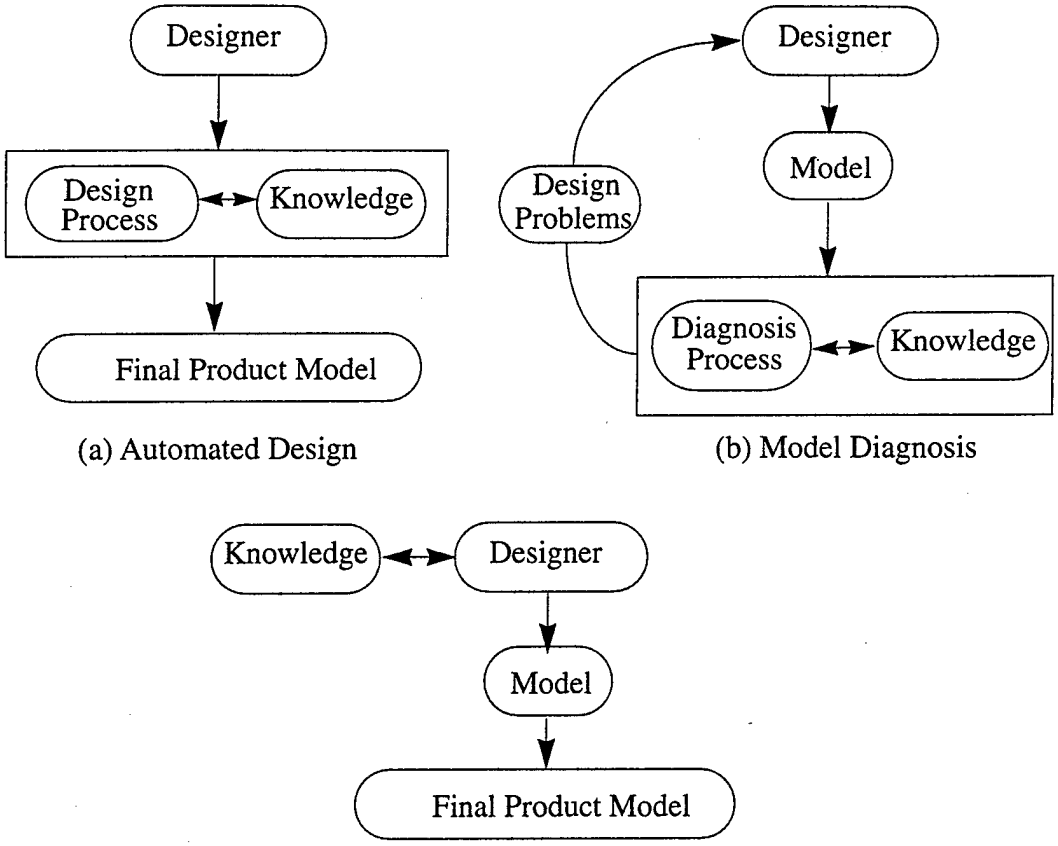
Several commercial systems have been developed to support the creation of AI based CAD applications, including ICAD,™ IntelliCorp's KEE™ and Wisdom Systems' Concept Modeller.™ The commercial systems provide sophisticated, interactive development environments for creating AI-based applications. However, they are only

support packages for knowledge-based system development, not solutions to any of the problems inherent in the methodologies.

2.2 Interactive Design Assistance

The AI methodologies cited above contributed significantly to the integration of engineering knowledge in CAD systems. As noted, nearly all of the systems targeted automated design or diagnostic applications. During the same time period, researchers also investigated the development of design systems that applied the same basic AI methodologies to integrate engineering design knowledge into interactive design systems. The difference between the automated and interactive design assistance philosophies has been the source of considerable debate. [MACC90] and [GALL95] argue that a fundamental problem with automated design systems is that they attempt to define rigidly a design process that is inherently dynamic and non-procedural. Although automated systems perform well in domains with rigid procedures and parameterized product models, any variances from the established regime are not allowed. Development of the CADRE system identified the weakness of attempting to define a "complete and correct knowledge base" for a design process that is intrinsically evolutionary [HUA96]. The designer and the computer must participate cooperatively during the design process, with the designer maintaining control of the process, identifying and presenting the problem while assuming full responsibility for the results.

Current design assistance methodologies can be categorized into three groups based on their designer interaction paradigms, as shown in Figure 2-4. The first interaction paradigm is that of the automated design systems. The design process and the knowledge are represented within the design system and operate to achieve a final design based on initial input from the designer. All of the automated systems presented in the previous sections operate with this style of interaction. In the second interaction paradigm, model



(a) Automated Design

(b) Model Diagnosis

(c) On-line Design Information

Figure 2-4. Designer Interaction Paradigms

diagnosis, the model is automatically diagnosed using internally represented analysis processes and design knowledge after the designer creates the model. The Component Design Advisor presented in [DIAZ94] employed the diagnosis style of interaction, analyzing a completed model and then advising the designer of any design critiques. The ProMod-S system described in [YEH96] operated on a similar premise for sheet metal manufacturing. The system analyzed, using a KBES, a completed geometric model provided by the designer to identify any difficulties that may be encountered during the manufacture of the part.

The PERSPECT system presented in [DUFF96b] approaches the problem of design support using the third interaction paradigm, on-line design information. The designer can query on-line design knowledge as shown in Figure 2-4(c) to explore similar past design cases within a particular domain. [OXMA93] reported a similar design assistant system that organized past designs into huge "design libraries" that respond to queries from the designer using case-based reasoning techniques. The resulting system supports the designer by allowing him or her to browse past designs and learn from those experiences. [CANT95] presents a design system that implements the same designer interaction philosophy, providing past design browsing support through a blackboard framework.

These systems implemented a variety of pro-active, or passive, design assistance that provided design advice after the designer completed the design process, before the process began, or in response to the designer's queries. The design assistance was not provided interactively to the designer as the solution developed. Many of the design environments required the designer to support the system's design process and respond to queries by the system for pertinent information. For example, the VEXED system presented in [STEI92] for VLSI design required the user to direct which design component to refine using a KBES. The VT system presented in [MARC88] implemented this type of user interaction and required the designer to provide key parameter values during the design process as a KBES shell refined the design solution.

2.3 Engineering Design Knowledge

It is widely accepted that the range of knowledge spanned during the design process is far broader than simple rules. [DIXO95] and [BRIN95] identify some of the varied knowledge types as:

- knowledge of the design artifact,
- design component relationships,

- facts and data,
- design processes, and
- analysis methods.

The development of a knowledge assisted design system must provide the means of representing this varied knowledge in a manner conducive to design support. However, as [BIJL87] explains, those representations innate to human designers cannot be directly translated to computer representations. While case-based reasoning may accurately reflect the manner by which humans reason about past design situations, experience shows the use of a single knowledge representation type in a design environment leads to the brittle system functionality experienced by the KBES.

To avoid the pitfalls of a single knowledge representation environment, [MASO95] presents an automated design system that utilized a blackboard framework with multiple, cooperative knowledge sources that represented varied engineering systems from process planning to cost analysis. [BAYL95] discusses a framework based on the research conducted for the ESPRIT project that effectively integrated multiple knowledge sources in an expert system developed to model the concurrent engineering philosophy. [DOWL94] supports the use of multiple, cooperative knowledge sources for representing varied engineering knowledge types, as can be found in concurrent engineering systems. The *Concept Designer* presented in [HAN95] implemented an object-oriented blackboard architecture that utilized multiple knowledge sources in support of chemical process synthesis. The knowledge sources were maintained separately to encapsulate their knowledge and functionality. By encapsulating the engineering design knowledge into separate, distinct sources, multiple types of knowledge can be represented in an integrated system. This also allows for the creation of knowledge sources responsible for single product components or processes [THOM95]. A similar approach can be found in [SOBO91].

The engineering knowledge sources utilized in an interactive design support system must also be able to communicate in a language compatible with the domain they represent. The concept of an ontology, outlined in [GRUB93], serves as a representational vocabulary for defining a domain. An ontology provides the means by which engineering knowledge can be exchanged between the system and the designer [DELO95]. Because of the complex nature of knowledge representations, the AI-based design systems required specialized knowledge engineers to code the knowledge for the system. [MAYE88] addressed this issue, noting that having an intermediary input the knowledge into the system led to distortions of the knowledge because of inconsistencies acquired from the knowledge engineer's misinterpretations of the expert's interview. Additionally, the systems needed to be debugged by the experts, but only the knowledge engineers could change the system.

2.4 Product Models

AI-based CAD systems that integrate an engineering product model with a knowledge resource in an object-based framework require the development of a model representation that must represent the full range of engineering information needed for the design process. The model does not have to include all of the possible engineering information about the design object, but it should contain the data from which the necessary and pertinent engineering information can be derived [KIMU95]. The product model is yet another means by which engineering knowledge can be introduced to the design system, and it represents what is being designed, and how it is designed, and its representation dictates the manner in which reasoning occurs [ROGE95]. [YAGI91] provides a historical perspective on model representations for CAD systems.

2.4.1 Model Decomposition

The vast amount of engineering information contained within any design model makes it necessary to decompose a complex design problem into smaller, more manageable segments. [KUSI95] provides a review of decomposing design artifacts and states that, for product decomposition, an artifact can be decomposed by structure and function. A structural decomposition separates the product into its various physical components and results in a hierarchical model of the geometric configuration of the design artifact. A functional decomposition separates the artifact based on the interactions of the various design components to perform the actions necessary to achieve the desired goals for that product [CHIT94]. An engineering object cannot be completely defined by either function or structure alone. The model must represent both the form and the function of the product to be of any practical use in a design environment that integrates engineering knowledge into the design process [GERO94].

[SCHE93] identifies the integration of conventional geometric models and the knowledge-based functional models as a fundamental problem in achieving so-called intelligent CAD systems. Their prototype system, however, only hints at the possibility of integrating the two models. Less than half of the representative systems presented thus far consider the geometry of the product. Those that consider part geometry provide assistance by diagnosing or analyzing a completed geometric model and do not provide assistance during the geometric construction process.

2.4.2 Features

The use of features for design integrates both the form and function of a design component in a single entity. Although the definition of a feature varies between applications, it is widely accepted that a feature is an encapsulation of engineering information. [BROW95] presents a feature-based model for a model-based geometric

reasoning system and defines a feature as "any perceived geometric or functional element or property of an object useful in understanding the function, behavior, or performance of the object." For example, after a functional decomposition has been performed on a design artifact, features can be used to represent the resulting entities of the model, which can be placed in a hierarchical model of the product. The power of the feature-based representation, though, is that the features represent the geometry of the part as well as other engineering information, such as tolerances and material properties. [DIXO87] and [SHAH95] provide reviews of the use of features in CAD.

Feature-based design (FBD) utilizes features as the building blocks to construct and represent a design artifact. Just as conventional CAD systems provide two-dimensional sketching and three dimensional solid modeling packages for geometric model construction, FBD systems provide pre-defined features for feature-based model construction. Unlike the geometric model, however, the feature-based model represents both the geometry and other engineering characteristics of the model. In addition, feature-based design implicitly builds design intent, or the rationale for making certain design decisions, into the final product model. Individual features represent certain functional aspects of the total design, and the choice of a particular feature in lieu of another represents the designer's intent [ROY95].

As the AI-based CAD systems have moved from the knowledge-based framework of the KBES to the object-based framework of the blackboard architectures, features have become a common basis for the integration of the engineering knowledge and models. [BATA93] discusses the benefits of the use of features for data and knowledge integration. The feature serves as yet another discrete form of knowledge about the complete product model to be used during the reasoning process, and, because feature-based models are hierarchical in nature, reasoning about the product can be performed on varied levels of detail. The use of features in the AI-based CAD systems, though, has been limited to

model diagnosis and critiquing environments that utilize a completed feature-based product model for a particular knowledge-based analysis [CHEN95], [BATA93], [BROW95], [BAYL95], [ROY95], [YEH96].

2.5 A Foundation for Knowledge Assisted Design

This chapter has presented the past efforts towards integrating experiential engineering knowledge in CAD systems. Many of the systems were developed primarily for automated design synthesis and model critiques and have shown limited success in those areas. While their model and knowledge representations do not support an open-ended, interactive design process, these past efforts do provide the fundamental building blocks upon which a knowledge-assisted deign methodology can be based.

The foundation for any CAD system is a representation of the design model that allows the design system to effectively perform its expected task. Conventional CAD systems are based on geometric models which can be constructed and analyzed using readily available design systems. As presented in Chapter One, the goal of this research is to develop a CAD methodology for an interactive design environment that integrates experiential engineering knowledge and conventional CAD tools and to provide interactive support during the design process. For this reason, the methodology must provide a proper representation of the design artifact, the design knowledge and the design process. These models must be integrated into a design system that provides interactive, opportunistic design support during the design process.

The next chapter presents the product, knowledge, and process models formulated for the knowledge assisted design methodology. These models build on a number of the approaches presented in this chapter and extend the representations to adequately support a dynamic, interactive design process. Chapter Four then presents the integration of the models into a framework for knowledge assisted design.

CHAPTER 3

REPRESENTING ENGINEERING KNOWLEDGE FOR DESIGN

As discussed in the previous chapter, the foundation for the knowledge assisted design methodology is appropriately representing the engineering knowledge for design. This includes representing the design artifact itself, the standards and guidelines that control and determine the attributes of the design artifact, and the design procedure for arriving at a final product model. The engineering knowledge representations build on the object-based system framework discussed in the previous chapter and extend the concept one step farther. The model is at the core of the methodology, and the design knowledge provides assistance to the designer during the design process. This chapter addresses both the model and the knowledge representations established for the knowledge assisted design methodology.

3.1 The Product Model

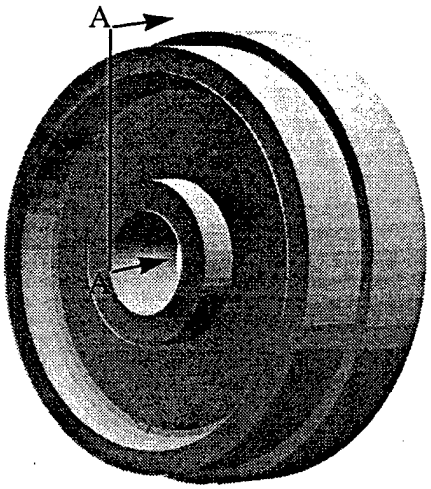
The product model structure is based on four fundamental requirements motivated by the shortcomings of the methodologies presented in Chapter One. First, the model structure must allow the designer to express both the functional and structural engineering content of the product, as well as design intent during the design process. Second, the model must represent the engineering content in a vocabulary familiar to the designer. Third, the knowledge must be able to interact with the model on various levels of detail. Finally, the model structure must be easily extensible, allowing the design engineer to

extend the model's structure without detailed knowledge of the design system's functionality.

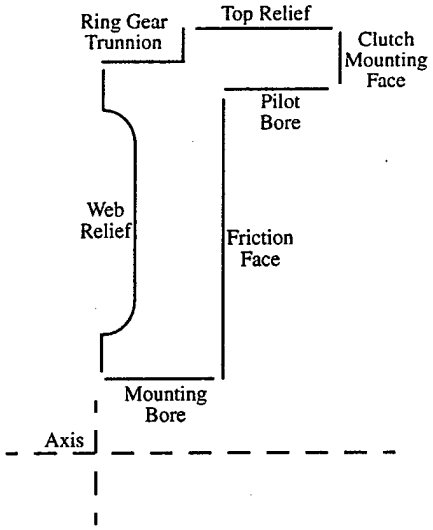
An object-oriented, feature-based model structure was formulated based on these requirements and preliminary investigations. The feature-based model was selected because features encapsulate both the form and function of the design artifact in single, distinct entities. A designer can perform both functional and geometric design simultaneously if features are used in conjunction with the feature-based design environment to be discussed in the next chapter. The dual functional and structural representation in features allows the designer to create a product model much richer in engineering content than the previous AI-based CAD systems that considered only functional properties of the design artifact and avoided geometry altogether. The feature-based model also decomposes a design artifact into smaller segments, thereby reducing a complex design problem to a number of smaller, more manageable problems. This was a key decision that laid the foundation for the development of the knowledge representation and design system functionality. The design example shown in Figure 3-1 presents a simplified model of the engine flywheel developed for the prototype design environment discussed in Chapter Five. The example will be used to provide an overview of the model hierarchy and introduce some of the key terms.

3.1.1 The Feature-Based Model Hierarchy

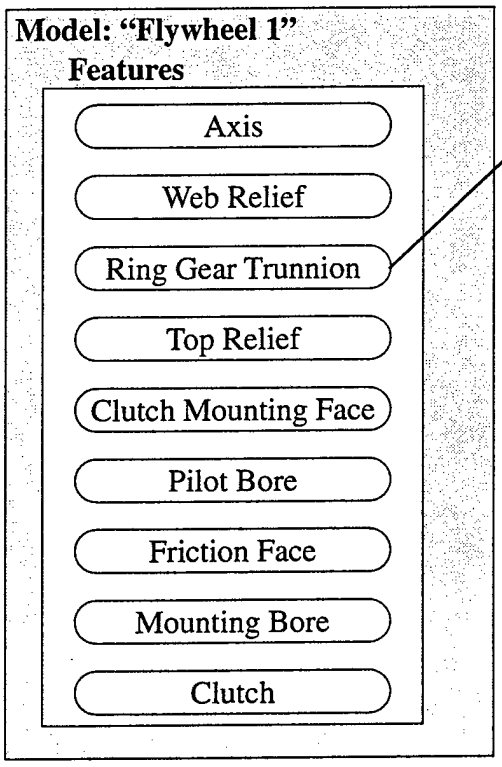
The model is organized in a hierarchical data structure shown in Figures 3-1(c) and (d) and consists of three layers. Each successive layer contains more detailed engineering information about the product. The outermost layer of encapsulation is the model, which presents the design as a single entity. Each instance of a model represents a unique design scenario and can be identified by its type and a unique identifier, which, for this example, are "flywheel" and 1, respectively. The model encapsulates the details of the product



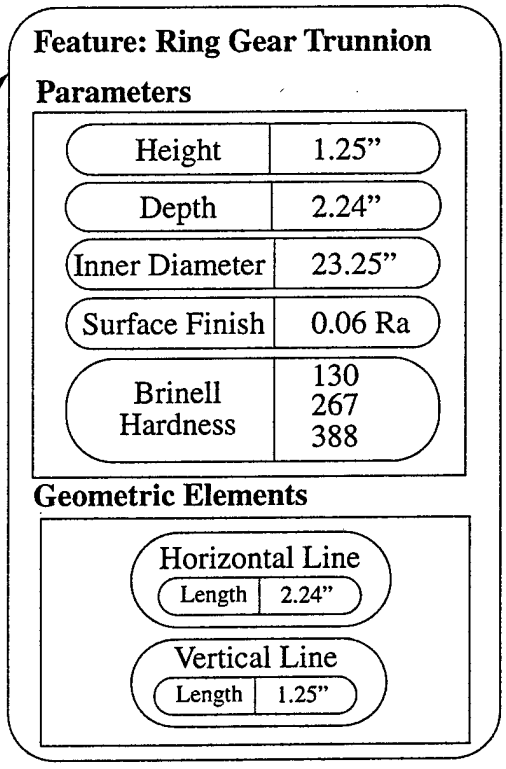
(a) Engine Flywheel



(b) Exploded View of Cross-Section A-A and Flywheel Features



(c) Flywheel Model



(d) Feature Parameters and Geometric Elements for the Ring Gear Trunnion

Figure 3-1. Flywheel Example of a Model Hierarchy

design as they are defined during the design process by serving as a container for, and providing external access to, the next layer of the hierarchy, features.

Features represent the functional components of the design artifact and are identified by the actual names of the entities used by designers to characterize their functionality. Features encapsulate the detailed geometric and non-geometric engineering model content in parameters and geometric elements which constitute the final layer of the hierarchy. Nine features comprise the flywheel model: axis, web relief, ring gear trunnion, top relief, clutch mounting face, pilot bore, friction face, and mounting bore.

Parameters are key-value pairs that represent the detailed geometric and non-geometric engineering attributes of a feature, such as dimensions and surface tolerances. The parameter key identifies the name of the attribute as defined in the designer's domain ontology. The ring gear trunnion feature, shown in expanded form in Figure 3-1(d), has five parameters identified by the keys height, depth, inner diameter, surface finish, and Brinell hardness. There are two types of parameter values: continuous and discrete. Continuous parameter values are defined in the feature as either a number or a string that the designer assigns a specific value during the design process. Discrete parameter values are defined in the feature as a set of values from which the designer can choose to assign the value of the parameter. The Brinell hardness parameter defines three discrete values from which the designer may choose during the design process. The other ring gear trunnion parameter values are continuous and may be assigned any value.

Geometric elements are the primitive entities that combine to define the form of a feature and are used to position the feature within the model. They are identified according to the designer's domain ontology, and each geometric element contains a list of parameters that define the entity's dimensions. For a two-dimensional geometric model, such as the flywheel cross-section, the geometric elements include lines and arcs. The ring gear trunnion geometry is defined by two geometric elements: a horizontal line and a

vertical line. Each line contains one parameter that defines the dimension "length" with a continuous value. Geometric elements are positioned with respect to one another based on the order in which they are added to the feature. Feature positioning will be discussed later in the chapter.

The product model structure satisfies the four requirements presented at the beginning of the chapter. Features encapsulate both the form and function of a design component, and when coupled with a feature-based design environment, allow the designer to express geometric and non-geometric engineering information and design intent, which satisfies the first requirement. The entities on all layers of the hierarchy are identified by the actual names used by designers, thereby satisfying the second requirement. The hierarchical structure of the model also allows the knowledge to interact with the model on increasing levels of detail, from the model to the parameters, and satisfies the third requirement. Finally, the object-oriented framework satisfies the fourth requirement. The designer only has to extend from the established framework to add functionality to the model, rather than reconfigure the entire model.

3.1.1.1 Feature Geometry

A more detailed feature hierarchy has been established to accommodate the various geometric configurations of features. Features descend from the parent feature class as shown in Figure 3-2. The next level of the hierarchy discerns the features with and without a geometric representation as geometric and non-geometric features. While the power of features lies in their capability to encapsulate both the form and function of a design component, features can represent non-geometric functional components of the model and are not required to contain geometric elements. The exploded view of the flywheel cross-section shown in Figure 3-1(b) identifies the eight geometric flywheel features by their

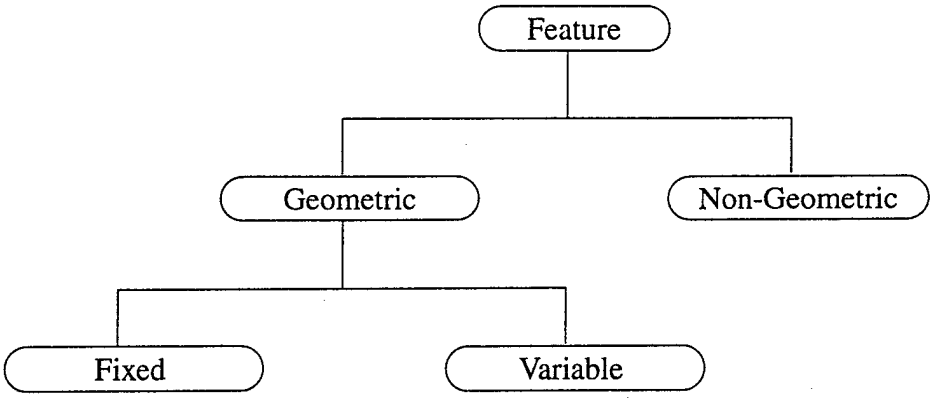
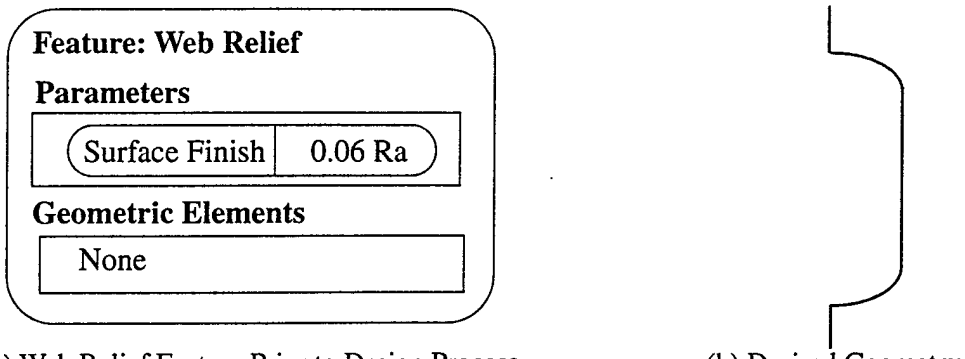


Figure 3-2. The Feature Hierarchy

location within the model. The ninth feature, the clutch, is a non-geometric flywheel feature. It contributes functionally to the design but has no geometric representation.

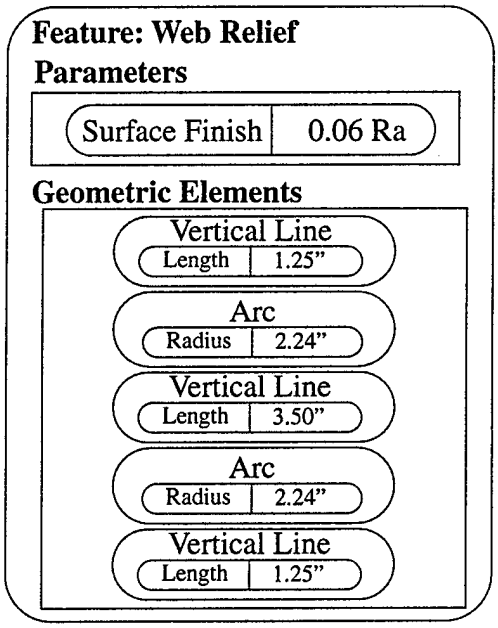
The hierarchy extends one level deeper to isolate the two geometric feature configurations: fixed and variable. Fixed geometric features are pre-defined with a set of geometric elements to describe the feature geometry. The geometric element parameter values can be modified during the design process; however, the type and number of geometric elements in the feature are immutable during the design process. The flywheel ring gear trunnion is an example of a fixed geometric feature. Its geometric elements are pre-defined, include a horizontal line and a vertical line, and cannot be modified during the design process.

Variable geometric features differ in that the number and types of geometric elements to define the feature geometry are mutable during the design process and are not pre-defined. The geometric element parameter values can also be modified during the design process. The flywheel web relief shown in Figure 3-3 is a variable geometric feature. Prior to the design process, the feature is defined with a name, web relief, a single parameter, surface finish, and no geometric elements as shown in Figure 3-3(a). During the design



(a) Web Relief Feature Prior to Design Process

(b) Desired Geometry



(c) Resulting Web Relief Feature Structure

Figure 3-3. Web Relief Example of a Variable Geometric Feature

process, the designer decides to define the web relief geometry as shown in Figure 3-3(b).Three vertical lines and two arcs are added to the web relief to achieve the desired geometry. The resulting feature is shown in Figure 3-3(c).

3.1.1.2 Feature Positioning

The feature positioning methods are based on the concept of "handles" presented in [ANDE86] and [BURC87]. Handles are attachment points defined within a feature's geometric elements that provide an object-oriented method for positioning geometric elements and features relative to one another. They are used to position geometric elements to define feature geometry and to position features to define the model geometry. Handles are identified by unique integers.

The handle positions for each two-dimensional geometric element are defined relative to one another in a planar cartesian coordinate system local to the geometric element. Consider the example of the horizontal line shown in Figure 3-4. The position of handle 1 in its local coordinate system is (0,0). The position of handle 2 is (l,0), where l represents the line's length parameter value. Similar positions are established for the vertical line and arc.

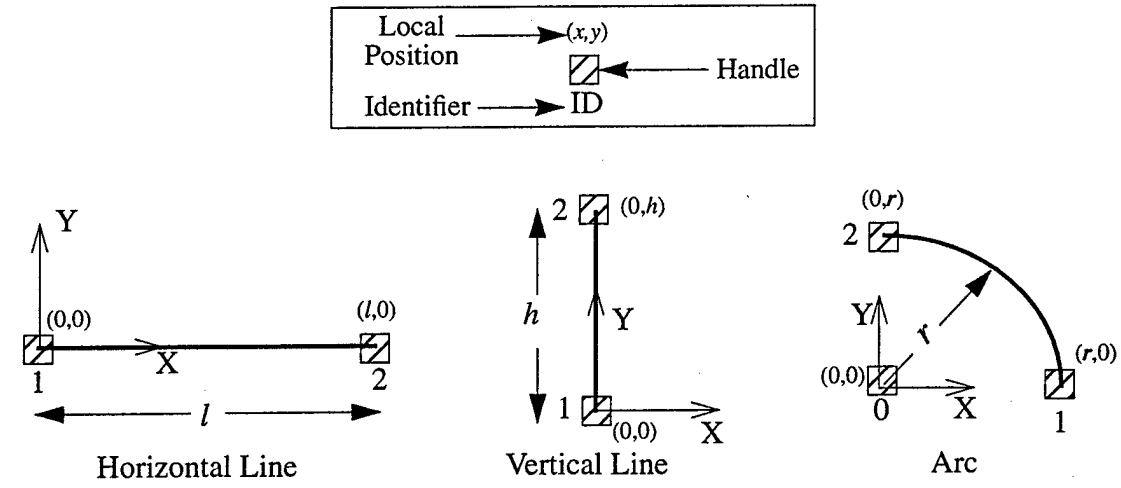
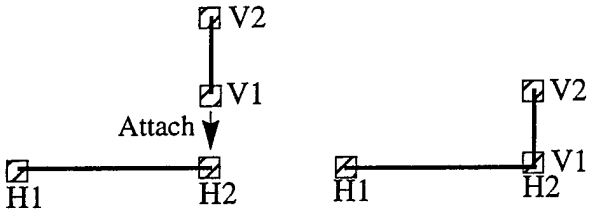


Figure 3-4. Feature Handles

Each geometric element is assigned a positioning relation that defines its location relative to another geometric element in the feature. The geometric elements developed for

the flywheel example and prototype design system are two-dimensional primitives that, by not allowing rotation, have two translational degrees of freedom; therefore, they must be positioned horizontally and vertically to establish their global position. Three positioning relations have been established to relate one geometric element with respect to another via their handles: Attach, HPos and VPos.

Consider once again the flywheel ring gear trunnion feature presented in Figure 3-1(d). The feature geometry is defined by two geometric elements: a horizontal line and a vertical line, shown in greater detail in Figure 3-5(a). Positioning the geometric elements within the feature demonstrates the first position relation, Attach. The Attach relation joins two geometric elements, thereby removing both degrees of freedom of the geometric element for which it is assigned. This also combines the two geometric elements into a

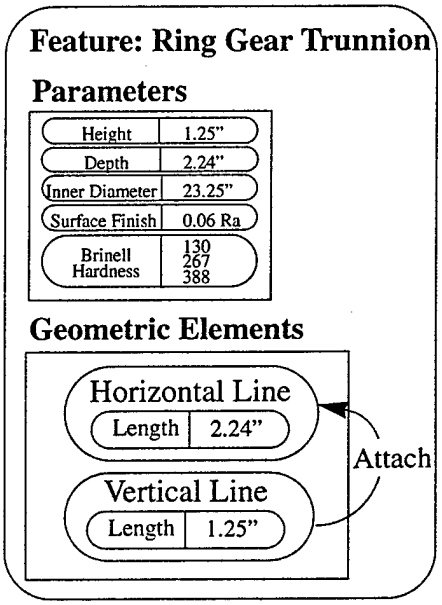


(a) Detailed View of Ring Gear Trunnion Geometric Elements and Resulting Geometry

Vertical Line

Attach (V1, Horizontal Line, H2)

(b) Vertical Line Attach Relation



(c) Positioning of Consecutive Geometric Elements

Figure 3-5. Positioning Geometric Elements Within Features

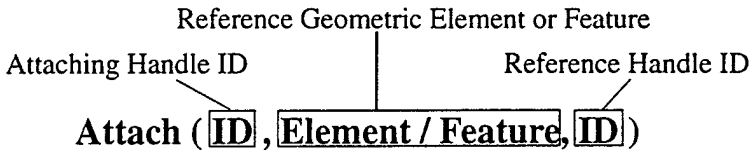


Figure 3-6. Attach Positioning Relation

single geometric entity. The format of the Attach relation is shown in Figure 3-6, and it includes a reference geometric element and feature because it can also be used to position features within the model. Geometric elements consecutively placed in a feature are positioned relative to one another using the Attach positioning relation, as shown in Figure 3-5(c). Once all the geometric elements have been added to the feature, it becomes a self-contained, fully constrained geometric entity. This is true for both fixed and variable geometric features.

While positioning geometric elements to define feature geometry only requires a single positioning relation, positioning features to define model geometry requires an extension of the feature hierarchy as shown in Figure 3-7. The extension isolates two feature types used in the feature positioning configurations: independent and dependent

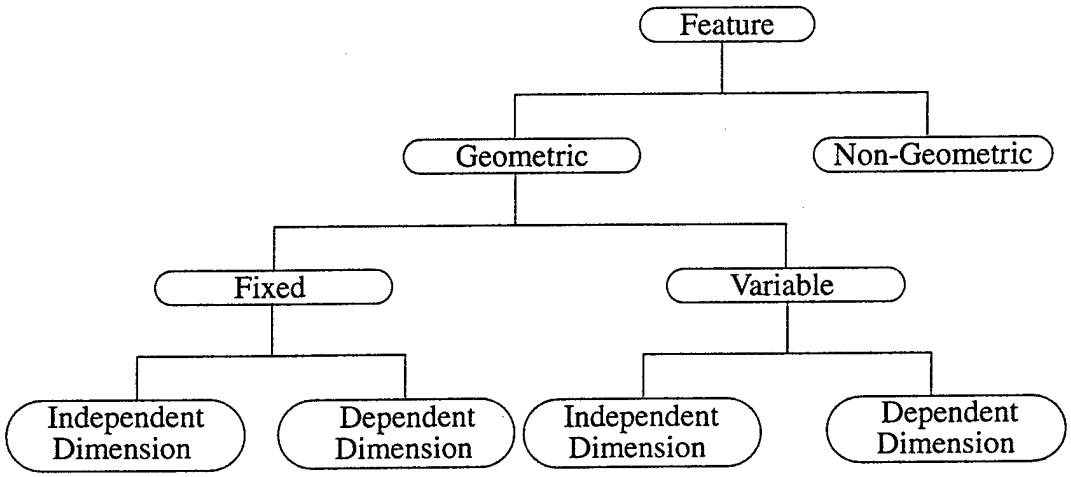


Figure 3-7. Positioning Extension of the Feature Hierarchy

dimensions. Independent dimension geometric features, or independent features, are features whose dimensions are not dependent on the positions or dimensions of other features. Independent features can have either fixed or variable geometry. Independent features are positioned using the Attach relation and HPos and VPos relations. The HPos and VPos relations assign the location of features relative to one another and remove either the horizontal or vertical translational degree of freedom, respectively, from the feature Additionally, the Hpos and VPos statements allow a feature to be offset from another by the value of the offset parameter. The format of both relations is shown in Figure 3-8.

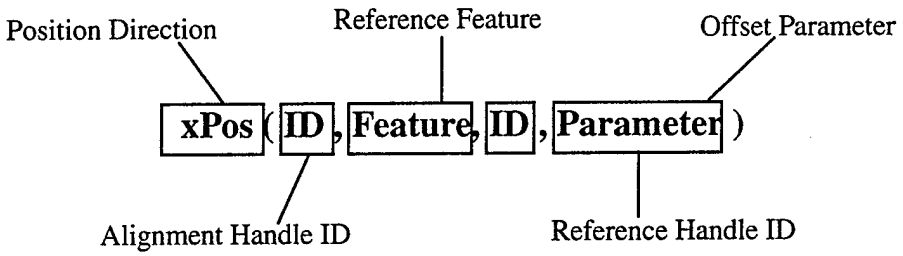
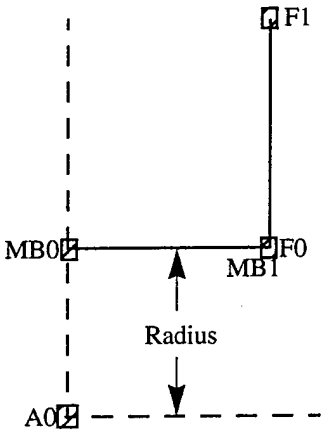


Figure 3-8. Horizontal and Vertical Positioning Relations Format

The example shown in Figure 3-9 illustrates the positioning of three independent features, the mounting bore, the clutch friction face, and the axis. The mounting bore is horizontally positioned by its MB0 handle, the alignment handle, with respect to the axis' A0 handle, the reference handle, using the HPos relation. The position is not offset, so the offset parameter is Null. The vertical position of the mounting bore is established using the VPos relation and is offset from the axis by the bore's radius parameter. The clutch friction face is positioned with respect to the mounting bore with the Attach relation which positions the clutch friction face's F0 handle is both horizontally and vertically with respect to the mounting bore's MB1 handle with no offset.

The effects of position and dimension changes on independent features are illustrated in Figure 3-9(c). Changes in their dimensions and positions only affect the position of the

feature that is related to them with a positioning relation. For example, the clutch friction face is attached to the mounting bore. If the position or dimension of the mounting bore changes, only the position of the friction face changes. The mounting bore is positioned with respect to the axis, so any changes in the position and dimension of the friction face do not affect the mounting bore's position.



(a) Positioned Features

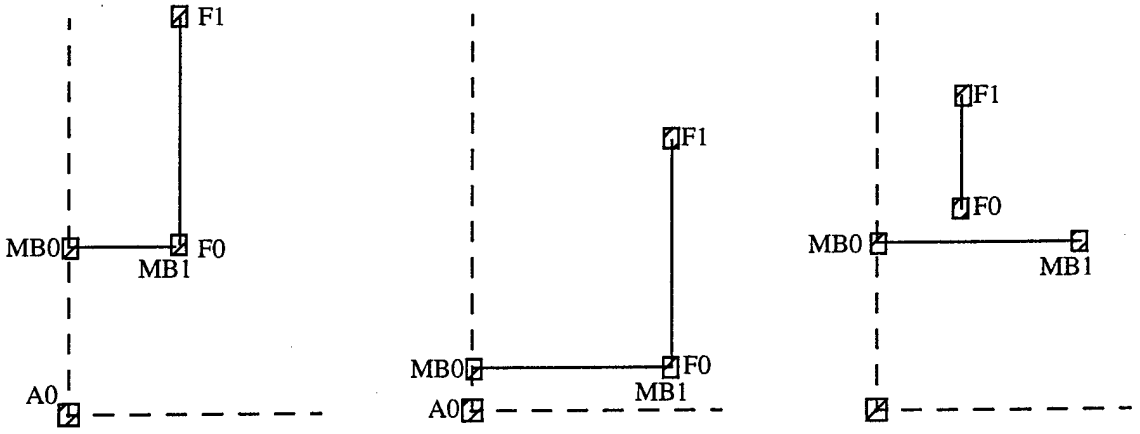
Clutch Friction Face

Attach (F0, Mounting Bore, MB1)

Mounting Bore

HPos (MB0, Axis, A0, Null)
VPos (MB0, Axis, A0, Radius)

(b) Positioning Relations



(c) Effects of Redimensioning and Repositioning Bore and Face

Figure 3-9. Positioning Geometric Entities for Independent Dimension Features

Dependent dimension geometric features, or dependent features, are features whose position and dimensions are dependent on both the positions and dimensions of independent features. Dependent features can also have either fixed or variable geometry. Positioning relations assign the location of geometric elements within the features, and dependency relations establish the feature's positioning and dimensioning relationships relative to other features. Each dependent feature has two dependency relations of the form shown in Figure 3-10. The dependency relations identify the independent features on which the feature is dependent and the handles from both the independent feature and the dependent feature to use for positioning. Two of the dependent feature's geometric elements must be labeled as having variable dimensions to accommodate the geometric changes that occur when changes in the position and the dimensions of either independent feature occur. One of the geometric element dimensions must be variable in the horizontal direction and the other must be variable in the vertical direction.

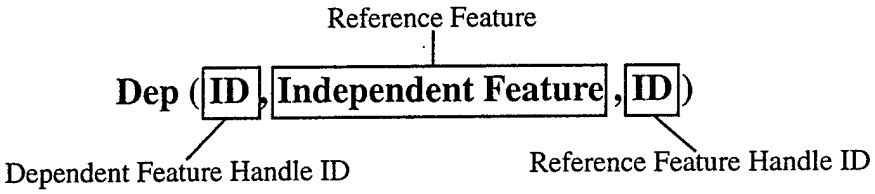
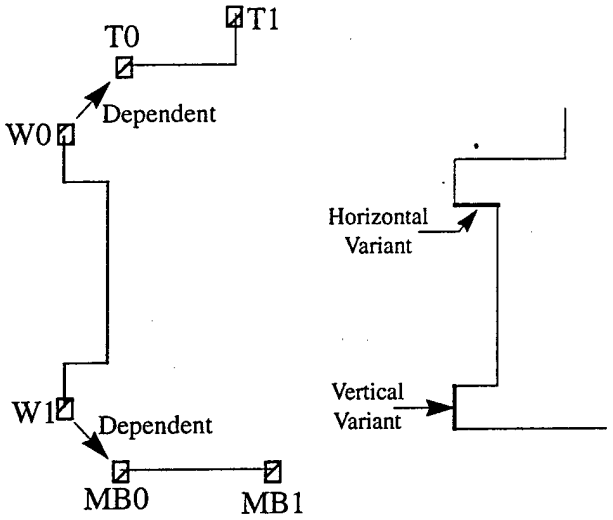


Figure 3-10. Dependency Declaration Format

The flywheel web relief feature is a dependent feature, and its position and dimensions are dependent on the ring gear trunnion and the mounting bore, as shown in Figure 3-11(a). The web relief's W0 handle is dependent on the ring gear trunnion's T0 handle, and its W1 handle is dependent on the mounting bore's MB0 handle, as defined by the dependency relations shown in Figure 3-11(b). Additionally, the horizontal and vertical variants have been identified and are shown in Figure 3-11(a). When the bore's length is increased and the ring gear trunnion is repositioned, the web relief maintains the

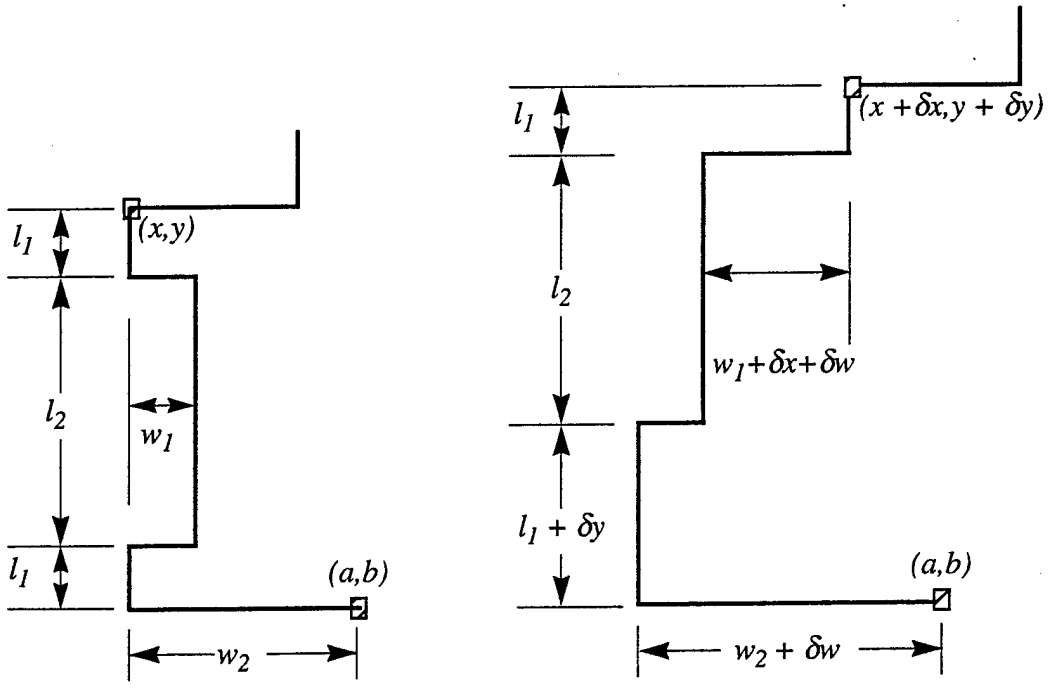


(a) Detail of Ring Gear Trunnion, Web Relief and Mounting Bore Positioning

Web Relief

Dep (W0, Ring Gear Trunnion, T0)
Dep (W1, Mounting Bore, MB0)

(a) Dependency Relations



(c) Effects of Redimensioning Bore Repositioning Trunnion

Figure 3-11. Web Relief Example of Dependent Feature Positioning

dependencies, as shown in Figure 3-11(c). The horizontal variant' s length increases by $\delta x+\delta w$ and the vertical variant's length increases by δy.

The design system and the designer can query a feature to determine the global position of any of its handles. The feature executes a method defined in its structure that recursively computes the global position of reference feature handles until the position is determined. The procedure for determining the global vertical position of a feature handle is shown in Figure 3-12. The horizontal position is determined in the same manner.

```
procedure determineVerticalPosition (handle)
    verticalPosition = 0.0
    verticalPosition = determineVerticalPosition (reference handleID)
    verticalPosition += vertical offset parameter value
    verticalPosition -= relative vertical position of attachment handle
    verticalPosition += relative vertical position of handleID
    return verticalPosition
end procedure
```

Figure 3-12. Pseudo-Code for Determining Global Handle Position

3.1.2 Featurizing

The process of decomposing a design product into features has been called featurizing [SHAH95]. Featurizing is very similar to performing a functional decomposition on a product; however, the functional components of the design are identified by geometry as well. The featurizing process for this methodology builds on that presented in [SHAH95] and provides an extension for identifying functional features that do not have a geometric representation. The following guidelines have been formulated to aid in featurizing a design product.

1. Examine drawings or CAD models of the product to be featurized, ensuring that exemplary versions and generations are considered.

2. Locate those areas of functional interest to the designer.

3. For each functional region, identify any areas that can be further decomposed by a particular sub-functional interest to the designer.

4. Isolate the geometry of each functional region, as well as the engineering parameters for the region. These are geometric features.

5. Determine the primitive geometric entities needed to build the functional features. These are the geometric elements.

6. Identify each geometric feature as positionally independent or dependent.

7. Identify any non-geometric functional components of the design and the engineering parameters for those components. These are the non-geometric features.

The featurizing process is likely to be an iterative process. The key to developing a feature base for a design product useful in this methodology is to ensure that the product is decomposed to the lowest level of functionality possible. The intent is to construct with and reason about features on the smallest grain of detail, thereby simplifying the entire design artifact into a number of minute encapsulations of engineering information, and to attempt to formulate a fundamental set of features that can be used to build all possible models in a domain.

3.2 The Knowledge

The knowledge representation is based on three main requirements motivated by the experience gained from the methodologies presented in Chapter Two. First, the representation must support the variety of knowledge used in the design process. The domain knowledge used for design covers a very broad spectrum, from simple rules to complex analysis procedures, and more than one of the types must usually be employed to arrive at an acceptable product design. Second, the representation must encapsulate the domain knowledge in a very narrow spectrum of applicability. Many of the past knowledge-based systems attempted to represent all of the domain knowledge in a single knowledge base which led to brittle systems that could not be extended easily. Finally, the

representation must be easily extensible. Domain knowledge should be able to be added to the knowledge base without detailed knowledge of the design system's functionality.

Based on these requirements and the knowledge representation used in the blackboard systems, an object-oriented knowledge representation has been developed for this methodology. The domain knowledge is encapsulated in multiple, autonomous knowledge sources that provide interactive support during the design process. The following sections detail the knowledge source structure and functionality, and describe the process for developing a set of domain knowledge sources. This knowledge representation has been used to incorporate SAE and ISO design standards in a prototype design system discussed in Chapter Five.

3.2.1 Knowledge Sources

The definition of a knowledge source is similar to that of a feature; a knowledge source is an encapsulation of engineering information. However, as features are encapsulations of the function and structure of the design product, knowledge sources are encapsulations of the design rules, and heuristic, analysis and support procedures of a product. Consider once again the flywheel's ring gear trunnion, and assume that the flywheel designer has, based on past experience, determined that the trunnion depth should be two times its width. The knowledge source for this rule would be of the form shown in Figure 3-13.

The knowledge source consists of three sections: the trigger conditions, the can-perform-action test, and the perform-action function. The trigger conditions identify the feature, or features, that the knowledge source will need to perform, and may modify during, its action process. The can-perform-action test is a preliminary test to determine if the knowledge source can apply its knowledge to the current model. The perform-action function is the actual application of the knowledge to a developing design solution. For this example, the ring gear trunnion is the only feature involved in the knowledge source's

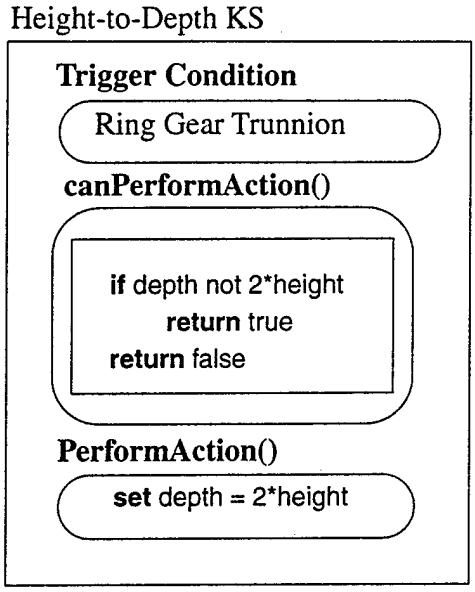


Figure 3-13. An Example Knowledge Source

action procedure. If the knowledge source is activated, it will first verify, using the can-perform-action test, that the depth parameter value is not twice the height parameter value. The perform-action function, which changes the depth parameter value accordingly, is called if the depth is not twice the height.

The form of all knowledge sources, regardless of their functional complexity, is the same as the previous example. Knowledge sources only have to define trigger conditions, the can-perform-action test, and the perform-action function to be utilized during the design process. The representation does not limit the type of engineering knowledge that can be used in the system. It allows for numerous types of knowledge from various design to be represented in the system, thereby satisfying the first and third requirements of the knowledge sources. The object-oriented structure also satisfies the second requirement, allowing knowledge on a small scale of granularity to be encapsulated within the knowledge source.

3.2.2 Developing Domain Knowledge Sources

The technique for developing the set of knowledge sources to represent the engineering knowledge for a design domain is similar to the featurizing process. Rather than decomposing the design product, however, the process of developing the domain knowledge sources decomposes the knowledge about designing the product to a set of specific design rules, advice, and procedures. Three important guidelines have been established for decomposing design knowledge and are summarized in the following list and explained below:

1. Limit the knowledge source applicability to the narrowest possible focus.
2. Knowledge sources must interact only with the model.
3. Limit the knowledge to design facts.

The first guideline addresses the problem of past knowledge-based systems representing all of the knowledge in a single knowledge base. The interactions and dependencies among the rules transformed the knowledge bases from a collection of rules to intricate programs. While later systems alleviated this problem by breaking the knowledge base into smaller knowledge sources, even the smaller knowledge sources encapsulated a large amount of information. The intent of the knowledge sources for this methodology is to represent domain knowledge decomposed to the smallest possible range of applicability, thereby simplifying the larger design problem into a series of smaller, more manageable solutions. The model structure is already decomposed to a very fine level of detail, and it facilitates focusing the knowledge on small pieces of information. The domain knowledge can be isolated to interact with a single parameter of a feature, the most detailed level of the model hierarchy.

This approach has three distinct benefits. First, narrowly focused knowledge sources are easier to develop and implement. A designer should be able develop and implement knowledge sources on site, which eliminates the need for a specialized knowledge

engineer to maintain the knowledge sources. Second, the knowledge base is comprised of a manageable set of discrete, autonomous knowledge sources, which eliminates the problem of complex, nested knowledge bases. Finally, it is easier to isolate faulty knowledge sources during the development process. Each knowledge source performs a single action. The designer can isolate the corresponding knowledge source and make the necessary adjustments if that process is not performing properly.

The second point is more an essential requirement than a guideline. Knowledge sources must interact only with the model. No interaction is permitted between knowledge sources. This alleviates the problems associated with complex, nested knowledge bases. If nesting is not allowed, the problems will not occur.

The third guideline establishes an important basis for this research. The knowledge sources provide design assistance to the designer during the model construction process. They are not intended to control the design process, mimic the designer's capabilities, or completely automate the design process. Most of the limitations of the past AI-based CAD systems resulted from their attempt to automate complex design processes. For this reason, knowledge sources should be limited to representing design facts that do not require complex, cognitive reasoning processes that may not arrive at a solution or cannot be represented in the computer. The knowledge sources should facilitate the design process, not model it.

Table 3-1. Knowledge Source Types

Fact Type	Action Description
Modification	Modify feature parameter values Establish positioning relationships Feature addition and removal
Advisory	Suggest possible design alternatives if precise modification is not possible
Analysis	Perform engineering analysis

During the course of this research, three types of knowledge sources have been established: modification, advisory and analysis. These types, shown in Table 3-1 with their respective action descriptions, provide a basis for establishing a set of domain knowledge sources.

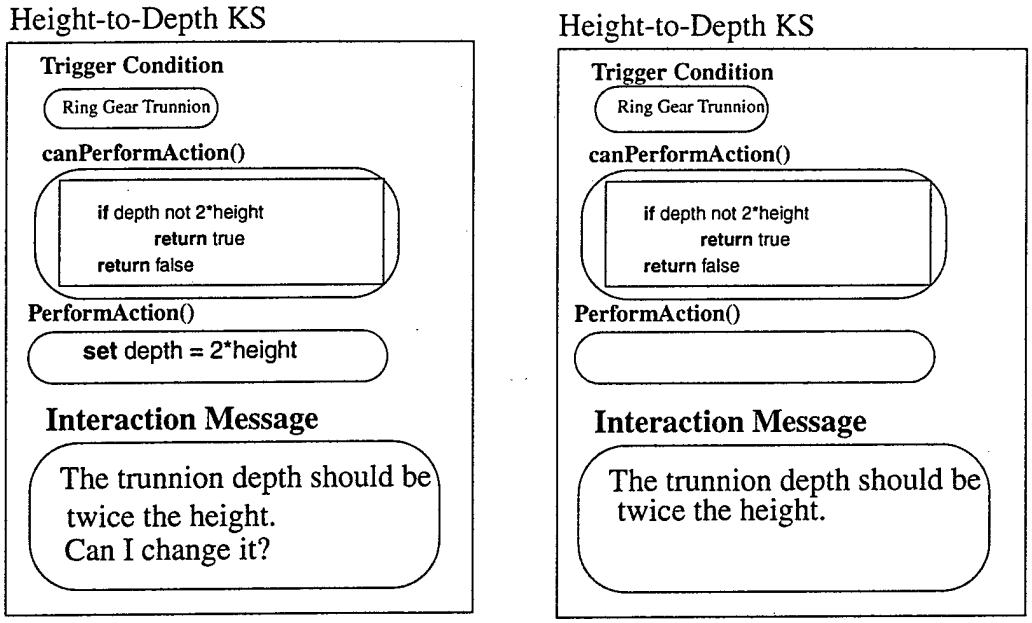
3.2.3 Designer Interaction

The principle purpose of the knowledge sources is to provide interactive assistance to the designer. Their advice, therefore, must be presented to the designer to consider the consequences of the recommended assistance. The presentation process can be implemented in two ways, interactive and automated. A knowledge source that implements the interactive approach presents its suggested assistance action to the designer, and the designer controls whether or not the knowledge source performs the action. A knowledge source that implements the automated approach performs its action automatically, presenting the assistance to the designer by actually modifying the model. The designer can then, if necessary, modify the changes made by the knowledge source.

3.2.3.1 Interactive

Knowledge sources that implement the interactive interaction approach are categorized into two classes based on the type of action they perform on the model: model modification and advisory. The first category includes knowledge sources that perform any type of model modification. The second category of knowledge sources only provide advice to the designer and do not perform any action on the model. An interaction message that explains the design advice to the designer must be defined for both types of interactive knowledge sources.

Consider the Height-to-Depth knowledge source presented in the previous section. Figure 3-14 shows two variations of the same knowledge source, illustrating the use of both interaction paradigms to provide an interactive assistance to the designer. The



(a) Model Modification (b) Advisory

Figure 3-14. Knowledge Source Dialog Definitions

knowledge source illustrated in Figure 3-14(a) implements the model modification paradigm. If the ring gear trunnion passes the can-perform-action test, the designer is prompted as to whether the knowledge source should perform its action and change the trunnion depth. The knowledge source shown in Figure 3-14(b), however, implements the advisory paradigm. The knowledge source does not perform a specific action; it only provides advice to the designer, recommending that the trunnion depth should be modified.

3.2.3.2 Automated

Knowledge sources the utilize the automated presentation approach do not require any additional methods defined within them. For example, the Height-to-Depth knowledge source shown in Figure 3-13 presents its action to the designer automatically. If the

knowledge source passes the can-perform-action test, the perform action method is immediately called to modify the model according to the rule it encapsulates.

CHAPTER 4

KNOWLEDGE ASSISTED DESIGN

The model and knowledge representations presented in the previous chapter provide the fundamental building blocks for developing the knowledge assisted design methodology. A complete implementation of the methodology, however, requires a proper integration of the knowledge, the model, and the designer. Many of the design assistant methodologies presented in Chapter Two integrate representations of the model, the knowledge, and the process in autonomous design environments and only allow the designer to support the design process controlled by the knowledge. The integration focus of this methodology is to bring the designer, the product model, and the domain knowledge together in an environment that allows the designer to dictate the process by which the product model is designed while receiving dynamic design assistance from the knowledge sources. This chapter discusses the resulting design environment and its functionality during the design process. The next chapter presents an example of the methodology applied to the design of engine flywheels.

4.1 The Knowledge Assisted Design Environment

The knowledge assisted design environment is an object-oriented framework based on the blackboard architecture. The framework utilizes an opportunistic problem solving environment with multiple, autonomous knowledge sources that provide interactive design assistance. The framework, shown in Figure 4-1, consists of two primary components: the

feature-based design environment and the design assistant. The designer interfaces with the system via the feature-based design environment and build the product model from a set of pre-defined, domain specific features. The design assistant serves as the intermediary between the knowledge sources, stored in a knowledge base, and the designer.

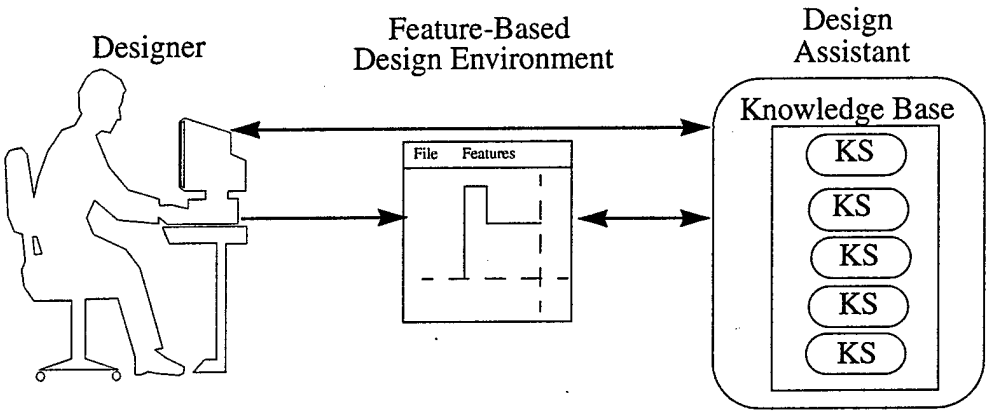


Figure 4-1. Framework for the Knowledge Assisted Design Environment

This framework integrates the designer, the model, and the knowledge to achieve the following properties of the knowledge assisted design methodology.

- The design process is controlled by the designer.
- The designer receives dynamic, interactive design assistance.
- The environment is domain independent and easily extensible.

Each system component encapsulates a distinct functionality contributing to the overall capabilities of the design environment. The following sections describe the system components, their functionality, and their contributions to these properties.

4.1.1 Feature-Based Design Environment

The feature-based design environment (FBDE) is the designer's primary interface with the system. It allows the designer to create, define, and edit a product model using the design-by-features paradigm for model construction. The FBDE provides the designer with a set of pre-defined, domain specific features that are used to create a product model. The features are stored in a feature base that is part of the FBDE.

The FBDE embodies two of the three methodology properties discussed above. First, it provides for a design process controlled by the designer. In fact, the domain knowledge is not required to complete the design process. The designer constructs the product model by adding features to the model and defining their parameter values and positioning relationships to arrive at a completed design. The knowledge is maintained separate form the FBDE and only supports model construction. The FBDE allows the designer to add features to and remove features from the model and modify the parameter values of individual features. In addition, the designer can select specific features and specify the positioning relations for those features. If the selected feature's geometry is variable, the designer can add geometric elements to the feature to define its geometry. The FBDE also supports multiple, concurrent product model construction during a single design session. The designer can construct multiple models to experiment with design variations and can switch between the product models during the design process. The multiple models can be from the same or different domains and are stored in a model within the FBDE.

The separation of the domain knowledge and the FBDE was serves two purposes. First, developing a design environment in which the designer controls the design process and the knowledge is not necessary to construct a product model achieves a knowledge-assisted design process. Although previous attempts at developing an interactive design assistant realized the benefits of maintaining the domain knowledge separate from the reasoning engine, the separation was only physical in nature. The function of the design

systems was so tightly linked to the domain knowledge that the system would fail if a complete representation of the domain knowledge was not defined. These systems were truly knowledge-based. In this methodology, only a partial representation of the domain knowledge is required because the knowledge is not directly linked to design completion. This alleviates the difficulty of representing complex design knowledge to achieve a completed design. If a particular aspect of the domain knowledge is very complex and difficult, or impossible, to accurately represent in a knowledge source, it can be left out of the domain knowledge set without depleting the system's capabilities. The domain knowledge sources are intended to facilitate the design process, and any complex cognitive reasoning knowledge should be left to the designer.

The second property the design environment embodies is the domain independence and extensibility of the methodology. The FBDE is specific to the domain of the features that are loaded into the feature base. The system can be extended for any design domain, provided a set of features can be defined for that domain and are loaded into the feature base. The complete domain independence of the methodology also requires that the domain dependent knowledge sources can be loaded into the system as easily as the features. This issue will be discussed in the next section.

4.1.2 Design Assistant

The design assistant provides the link between the domain specific knowledge sources, the designer, and the product model. It contains a knowledge base that is similar to the feature base in the FDBE. The knowledge base stores the knowledge sources for the specific domain in which the designer is designing. One design assistant is instantiated for each model created in the FBDE; therefore, several can be in operation during each design session.

The design assistant performs three distinct functions within the knowledge assisted design environment. First, the design assistant dynamically respond to modifications in the model and presents the changes to the knowledge sources. The design assistant presents three types of model modifications to the knowledge sources:

- feature addition and removal,
- feature parameter value modification, and
- feature positioning modification.

When the design assistant presents the model modifications to the knowledge sources, those knowledge sources with a trigger condition that matches the type of the modified feature are grouped in a queue of applicable knowledge sources. The design assistant then performs its second function, which presents those applicable knowledge sources to the designer. The manner in which the knowledge sources are presented to the designer will be discussed in the next section. The final function of the design assistant is to maintain a clear separation and mediate interaction between the knowledge sources and the model. Neither the knowledge and the model, nor the knowledge and the designer, can interact without the design assistant.

The design assistant substantiates the final characteristics of the knowledge assisted design methodology. It provides dynamic, interactive design support by way of the knowledge sources. The design assistance is presented to the designer opportunistically during the design process. The knowledge sources do not present themselves in a pre-defined order or at a pre-defined time. They react to the design process controlled by the designer, who can interact with the knowledge sources and affect how they are applied to the current model.

The methodology also achieves complete domain independence because of the design assistant's framework. Changing the domain in which the design assistant operates is as simple as loading a new set of knowledge sources into the knowledge base. The

functionality of the design assistant is independent of the type of knowledge sources contained within the knowledge base. The design environment is capable of supporting any domain that is properly defined according to the proper format, provided that the domain of the knowledge sources matches that of the feature base.

4.2 System Interactions During the Design Process

Having established the design environment framework and the basic functionality of its components, a detailed examination of the design process is necessary to completely understand the interactions taking place between the components within the system during a design session. The design process is controlled by the designer, and there are no rules built into the system that control how the it proceeds. However, because the designer process takes place in a FBDE, the design process usually proceeds as follows. First, a new model is created. Individual features are then added to the model and their parameter values are defined. As more features are placed in the model, the inter-feature positioning relations are defined, and knowledge source modification actions further refine the feature parameter values. A complete model is achieved once all the necessary features have been added to the model, their parameter values have been properly defined, and they have been positioned within the model to form a complete geometric model. Both the designer and the design assistant contribute to the final design throughout the design process. It must be emphasized, however, that the designer has the final say in any changes that occur to the model.

4.2.1 Instantiating a Model

At the onset of a design session, the FBDE cannot be used to construct a product model until the designer instantiates, or creates, a new model. The process of instantiating a new model triggers a series of important actions within the design environment that

initialize the system for the design process. To create a new model, the designer must specify the domain to which the model belongs. The system then dynamically loads the domain feature set into the FBDE's feature base. If the features are successfully loaded, the system initializes the model hierarchy. The FBDE assigns the model a name, which corresponds to the domain to which it belongs, and a unique identifier which differentiates it from other models created within the same domain. The model hierarchy initially contains a model layer with an empty feature list. The FBDE places the new model in its model list.

Once the base of the model hierarchy is established, the system creates a design assistant for the new model and initializes it with a reference to the model and an empty knowledge base. The system also initializes the design assistant to observe the product model and recognize any modifications that occur in it during the design process. The design assistant then dynamically loads the domain specific knowledge sources into the knowledge base, just as the features were loaded into the feature base. The system is now completely initialized, and the designer can begin the design process.

The instantiation process differs slightly if a new model is created after the design environment has been initialized. If a model of the same domain as the new model already exists, the initialization process creates a new model hierarchy and adds that model to the FBDE's model list. If the new model's domain is different, the initialization process proceeds as follows. First, the FBDE creates a new feature base is created and loads the new domain feature set into it. Second, the FBDE creates a new model hierarchy and adds it to the model list. The FBDE also instantiates a new design assistant and, the design assistant loads the domain specific knowledge sources are adds them to its knowledge base. The designer can now use the design system to design multiple models by simply switching between them during the design process.

4.2.2 Building the Model

The process of building and defining the product model begins once the designer creates the model and the system initializes the design environment. Model construction is decomposed into two distinct stages of feature modification: start-up feature modification and feature modification. Each process involves adding features to the model and modifying their form or function.

4.2.2.1 Start-Up Feature Modification

The start-up feature modification process occurs immediately following model instantiation and design environment initialization. At that time, the FBDE prompts the designer to define the parameter values for a set of start-up features. Start-up features are functional features that define the initial input specifications for the product model. This set of features is pre-defined and maintained in a file that lists the feature types. For example, if the clutch is considered a start-up feature for the flywheel presented in Chapter Three, the file will consist of one line that specifies *engine type* as the feature type.

This stage of the design process is similar to the designer providing the input specifications for the automated design systems. The designer is not required, however, to define the start-up feature parameter values for the design process to continue. When prompted, the designer can either define the start-up feature parameter values or cancel the action. If the designer chooses to cancel the action, the design process proceeds directly to feature modification. If the designer defines the start-up feature parameter values, however, those features are added to the model. The resulting internal actions will be discussed in the next section. This process also differs from providing input specifications for the automated design systems because it is not necessary to provide any start-up features for a particular domain. If the file is empty, the design process simply continues to the next phase.

The following guidelines have been established to assist in determining which features to include in the set of start-up features for a particular design domain.

- Include features that define the design limits of the product, both functionally and geometrically.
- Include features on which many or all of the features in the model are dependent.
- Do not include features that are dependent on several other features in the model.
- Include only functional features that must be present in every variation of the model.

Although the inclusion of start-up features is optional, it provides a valuable type of assistance to the designer and defines the core foundation and design envelope for designing the product model.

4.2.2.2 Feature Modification

The feature modification stage constitutes the bulk of the design process and includes adding features to the model, removing them from the model and defining their parameter values and geometric elements. Unlike the start-up feature modification stage, however, the designer is not prompted to define a specific set of features and add them to the model. The designer controls the feature addition process and decides when to add a particular feature and define its parameter values. The designer can also modify or remove features previously added to the model and define the positioning relations for those features that have a geometric representation. Each of these actions modifies the model, and the design assistant observes them as changes in the model and triggers a series of events within the knowledge assisted design environment.The events triggered within the FBDE when a model modification occurs update the status of the model. For example, when the designer selects a feature to edit, the FBDE presents the feature's parameters to the designer and applies any changes the designer makes to the current model.

The events triggered outside the FBDE when a design modification occurs provide the dynamic assistance to the designer during the design process. The ensuing events are

independent of the type of modification to the model. When a modification occurs, the model notifies the design assistant that a particular feature has been modified. During the notification process, the design assistant establishes a reference to both the model and the modified feature and presents them to the knowledge sources. The design assistant then queries each knowledge source to determine if it may be able to provide design assistance based on the type of feature modified. During the query, each knowledge source compares the modified feature type to those listed in its trigger conditions. If the types match, the knowledge source responds affirmatively to the query. If they do not match, the knowledge source responds falsely, and the design assistant queries the next knowledge source in the knowledge base.

The knowledge sources that respond affirmatively to the design assistant's query are placed in an action queue. The queue sorts the knowledge sources based on the number of trigger conditions in each knowledge source, which isolates those knowledge sources that perform an action on the model that involves only on the modified feature from those that utilize several features for design assistance. The knowledge sources are sorted to reduce the possibility of potential conflicts, an issue to be discussed in Section 4.3. Following the queue sorting process, the design assistant attempts to present the knowledge sources one at a time to the designer. First, the design assistant queries the knowledge source's can-perform-action test. If the knowledge source responds falsely, the query is performed on the next knowledge source in the action queue. If the knowledge source response is true, then it presents its action to the designer.

The knowledge sources present their design assistance to the designer using either the automated or the interactive approach. Knowledge sources that implement the automated interaction approach immediately perform their action. Knowledge sources that implement the interactive presentation approach present their assistance to the designer to control whether the knowledge source applies itself to the model. If the advice is a

recommendation and does not modify the model, the knowledge source presents the design advice and the design process continues. If the knowledge source can provide assistance by modifying the model, the designer must choose whether or not to allow the knowledge source to perform the modification. If the designer allows the process to occur, the knowledge source immediately performs its modification action. If the designer chooses not to allow the action to be performed, however, he or she must provide an explanation for contradicting the suggested design modification. The system stores the explanation with the model data which can be viewed at a later time to justify any design decisions that may violate established design standards.

Once a knowledge source has performed a modification action on the model, the model notifies the design assistant that a change has been made, just as it did when the designer modified the model. The knowledge sources are queried again, and the design assistant creates and sorts a new list of applicable knowledge sources. This process continues each time a knowledge source modifies the model, building a tree of possibly applicable knowledge sources that are traversed in a depth first manner, as shown in Figure 4-2, until all the nodes have been visited.

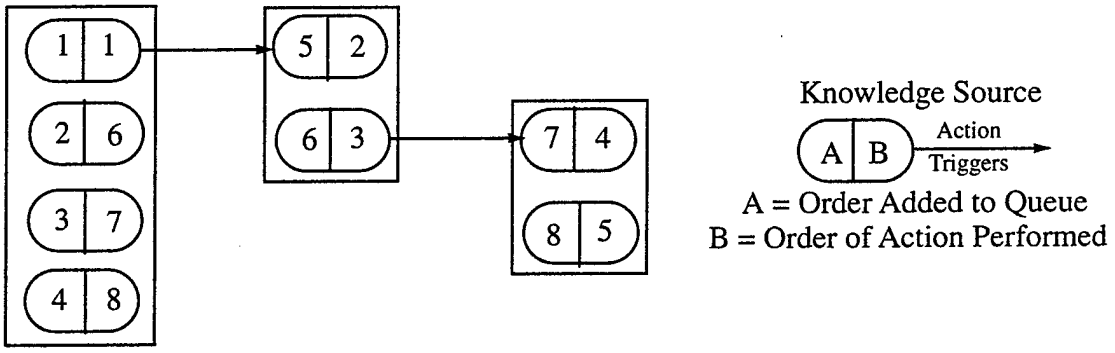


Figure 4-2. Depth First Knowledge Source Action Traversal

During the feature modification design stage, the designer also defines the geometry for features with variable geometry. This process includes adding and removing geometric elements and defining their dimension parameter values and positioning declarations. Adding geometric elements to a feature is equivalent to adding features to the model. The feature stores geometric elements in a list as they are added by the designer. By default, Attach position relations are assigned to position the geometric elements with respect to one another. Consider the example shown in Figure 4-3. Feature B is an independent feature with variable geometry. When the designer adds Feature B to the model, the system initializes the feature with an empty geometric element list and a single handle, BO. To achieve the desired feature geometry, as shown in Figure 4-3(a), the designer first adds as arc and a vertical line to the feature. When the arc is added, its A0 handle

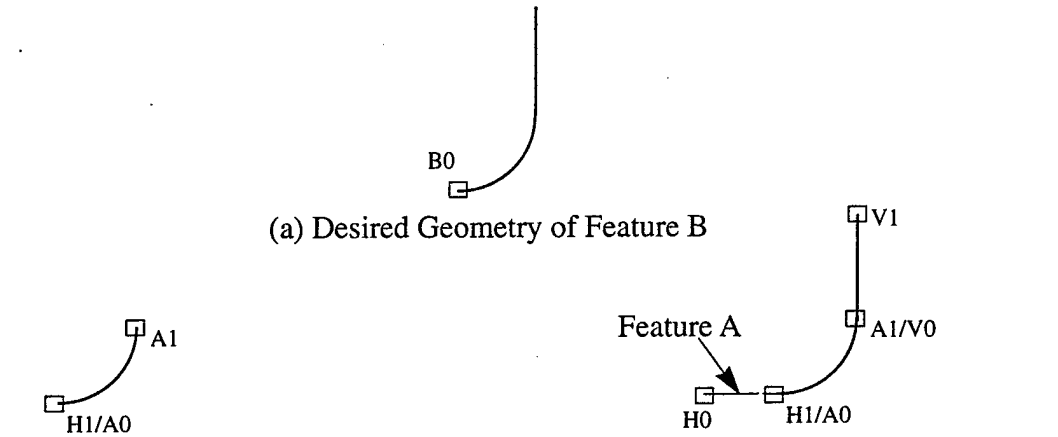
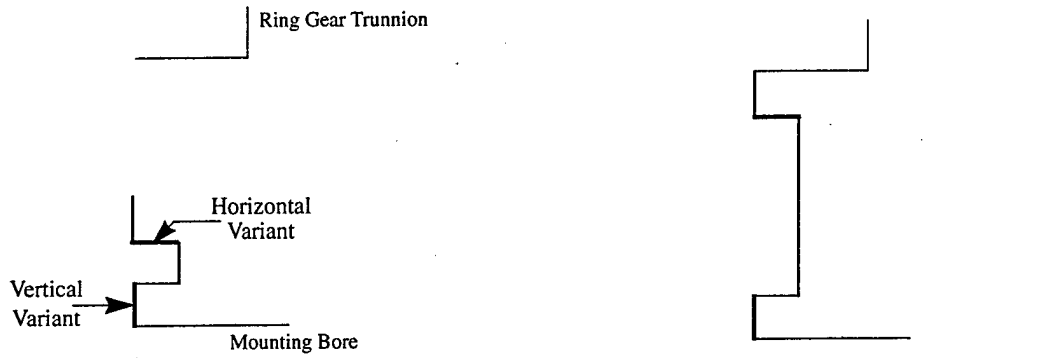


Figure 4-3. Defining Variable Geometry

automatically replaces the B0 handle. When the vertical line is added, the system defines an Attach relation that positions the vertical line's V0 handle with respect to the arc's A1 handle. Once the designer completely defines the desired geometry for Feature B, he or

she assigns an Attach position relation between Feature A's H1 handle and Feature B's A0 handle to achieve the geometry shown in Figure 4-3(c).

The designer can define the geometry of an independent feature, as illustrated in the previous example, once he or she adds the feature to the model. If the position is dependent, however, the designer can define the feature's geometry only after the dependency relations have been assigned to the feature. Additionally, a horizontal and vertical variant geometric element must be specified before the editing process is complete.



(a) Defining Web Relief Geometry (b) Resulting Constrained Geometry

Figure 4-4. Web Relief Example for Defining Feature Geometry

Consider the example shown in Figure 4-3 which illustrates the process of defining the geometry for the flywheel web relief presented in Figure 3-11. To achieve the desired geometry shown in Figure 4-3(b), the designer or the knowledge sources must first assign the dependency relations that relate the mounting bore and the ring gear trunnion to the web relief. Once the relations have been assigned, the designer adds three vertical lines and two horizontal lines with their default dimensions, which the system adds to the feature and positions with Attach relations, resulting in the geometry shown in Figure 4-

3(a). The designer assigns the horizontal and vertical variants as shown, and, upon completion of the geometry specification process, the system constrains the geometry to connect the bore to the trunnion. The model then notifies the design assistant of a model modification, and the design assistant queries the knowledge sources as discussed in the previous section.

4.3 Design Process Control

Although the design process is externally controlled by the designer, the knowledge assisted design system implements internal control procedures to avoid conflicts and loops associated with knowledge source actions. The first procedure prevents the development of loops when the design assistant places the applicable knowledge sources in the action queue. As shown in Figure 4-4, if the first knowledge source in the action queue is

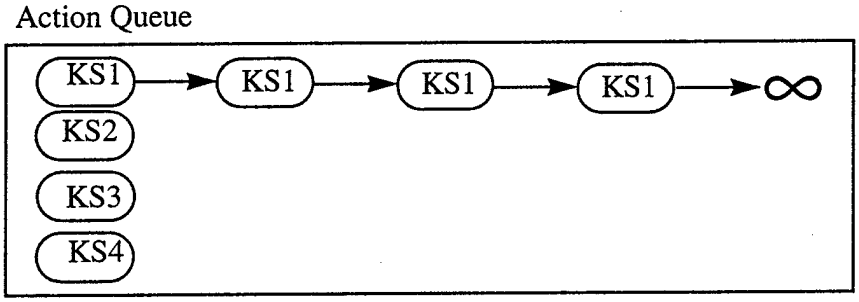


Figure 4-5. Infinite Action Loop

triggered by, and modifies, a particular feature, the model will notify the design assistant of the change, and the design assistant will place the same knowledge source as the first knowledge source in the second level of the action queue tree structure. This process will cycle indefinitely without some form of control. To avoid this problem, the system places a

lock on each knowledge source as it performs its action, which prevents the design assistant from querying it. While the knowledge source's action may trigger additional knowledge sources to be placed in the queue, once a lock is placed on it, the design assistant cannot add the knowledge source to the action queue again until its action is complete and the design assistant removes the lock.

The second internal control procedure deals with the automatic presentation of knowledge source actions. Consider the example of the automated flywheel rim Height-to-Depth knowledge source presented in Chapter Three. As an automated knowledge source, the designer cannot control its application to changes in the model. If the designer chooses to set the depth to a value other than two times the height, the knowledge source will automatically change the value back. Without any for of control, the designer will never be able to change the depth to any other value. To avoid this problem, the designer is prompted to intervene if an automated knowledge source is applied more than once during the same design session. The designer can either disable the knowledge source and provide an explanation for the deviation or leave the knowledge source enabled and accept the enforcement of the rule. A disabled knowledge source cannot contribute to the design process until the designer specifically enables it.

The designer also has two means of manipulating the applicability of interactive knowledge sources as they are presented during the design process. First, the designer can allow a knowledge source to apply itself to the model and then disable its interactive presentation capability. The doing so, the designer accepts the knowledge source's action and allows the knowledge source to apply itself automatically for the remainder of the design process. Second, the designer can disable a knowledge source by rejecting its recommended action. In this case, the designer is prompted to provide an explanation for the deviation, and the knowledge source is disabled. Disabled knowledge sources may be enabled by the designer at any time during the design process.

A final form of knowledge source control allows the designer to disable all of the knowledge sources' participation in the design process. The knowledge sources can be completely disabled if the designer wishes to build a product model free from any interaction with the knowledge sources. While this defeats the purpose of a knowledge assisted design environment, it allows the designer to maintain complete control of the design process. The designer can re-enable the disabled knowledge sources at any time during the design process.

CHAPTER 5

AN APPLIACTION OF KNOWLEDGE ASSISTED DESIGN FOR FLYWHEELS

A prototype knowledge assisted design system has been implemented to validate the design methodology presented in Chapters Three and Four. The design system is a domain independent framework that allows the designer to create a product model in a feature-based design environment and receive design support from multiple, autonomous knowledge sources via the interactive design assistant. This chapter presents the design environment and the system capabilities during a typical design session when applied to the design domain of engine flywheels, for which a set of features and knowledge sources were developed.

5.1 The Design Environment

The prototype knowledge assisted design environment is a domain independent design system that can be used for knowledge assisted product design, provided a set of features and knowledge sources are defined for a specific domain. The system was written in the Java programming language which was chosen because of its object-oriented framework and numerous built-in capabilities, such as platform independent graphics, string manipulation, and networking. The design system was tested on Silicon Graphics and Sun workstations, as well as Window's PCs, all using the same compiled code, a distinct benefit of the Java programming language.

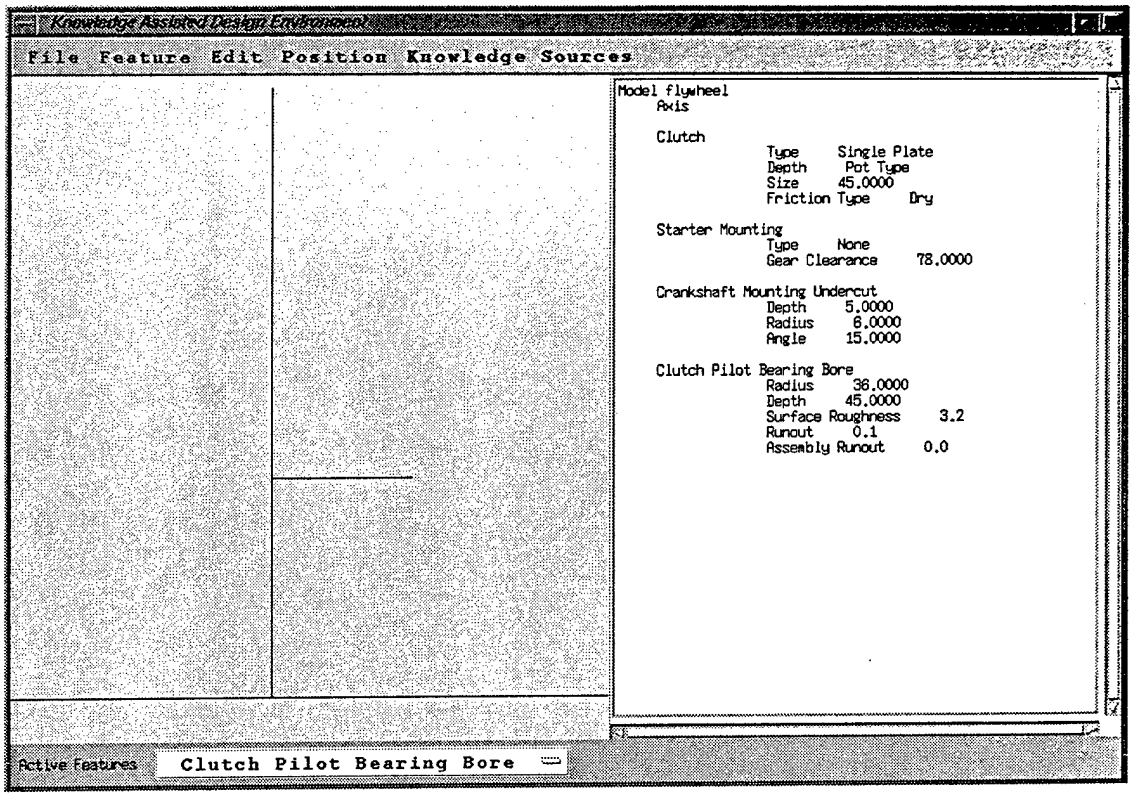


Figure 5-1. The Graphical User Interface

The designer interfaces with the knowledge assisted design system using the graphical user interface (GUI) of the feature-based design environment shown in Figure 5-1. The GUI consists of three primary components. The first component is the menu bar that provides the designer all of the necessary commands for building a complete model. The designer may add features to the model, remove features from the model, edit the parameters and geometry of individual features, position features within the model, and enable and disable the knowledge sources. The second component of the design environment is the model visualization panel. This panel provides a dual graphical and textual representation of the model during the design process. The graphical representation displays the two-dimensional geometry of the features and their positions within the model. The designer can interactively pan and zoom the model view and select

individual features within the graphical panel. The textual representation presents each feature and its respective parameters and parameter values throughout the design process, which allows the designer to visualize both the non-geometric features of the model and the features that have not been positioned within the model. The third component of the design environment is the active feature choice selection, which allows the designer to select a particular feature to modify. The selection choice allows the designer to select those features that do not have a geometric representation or have not yet been presented graphically to the designer.

The other components of the knowledge assisted design system, including the design assistant, the knowledge base, the model, the knowledge sources, and the features have also been implemented according to the framework established in the previous chapter and are written in the Java language. The remainder of this chapter discusses the application of the knowledge assisted design methodology to the design of engine flywheels, the features and knowledge sources developed for that purpose, and the designer-system interactions during a typical design session.

5.2 The Design Domain

The specific domain to which the knowledge assisted design methodology has been applied is the design of engine flywheels. An engine flywheel provides the mass and inertia necessary to minimize the fluctuations of the engine speed during load changes. The flywheel also provides the mounting surface for the ring gear used during engine starting, as well as the mounting surface, clearance, and driving face for the clutch. The features and the domain knowledge sources used for designing flywheels were developed in collaboration with the Cummins Engine Company according to the guidelines and procedures discussed in the previous chapters.

5.2.1 Flywheel Features

The flywheel features were identified using the featurizing process. Drawings and designs of flywheels from over fifteen different engine families and five different clutch variations were examined to identify the features necessary to design flywheels. The geometry was simplified to a two dimensional cross-sectional representation because of the axi-symmetric nature of the flywheel geometry. From this process, thirty-five features were identified and are shown in Figure 5-2. Of those features, two are non-geometric, the clutch and the starter mounting. Four geometric elements were also identified: horizontal lines, vertical lines, angled lines, and arcs.

Once the features of the flywheel were established, their engineering parameters were defined. The parameters for the flywheel features range from dimensions and tolerances to type identifiers, which implement both discrete and continuous value types. For example, the engine family feature has two parameters. The first, a discrete value type, allows the designer to identify the engine type from a pre-defined set of engine families. The second parameter of the engine family, the distance to the crankshaft mounting face, has a continuous parameter value type that identifies the distance from the engine block to the flywheel crankshaft mounting face.

The final process of defining the flywheel feature set was identifying the positioning relations and geometric variability among the features. Four of the flywheel's features have dependent positioning properties: the web relief, the top relief, the clutch face relief, and the crankshaft mounting face. Of those dependent features, the crankshaft mounting face has fixed geometry, a single vertical line, and the others have variable geometry. The remaining flywheel features have independent positioning and fixed geometry patterns. None of the independent features were characterized with variable geometry.

1 - Axis
2 - Clutch Pilot Bearing Bore
3 - Pilot Bearing Bore Chamfer
4 - Pilot Bearing Bore Hub
5 - Clutch Face Relief
6 - Spring Pocket Bore
7 - Spring Pocket Bore Chamfer
8 - Clutch Friction Face
9 - Face to Bore Fillet
10 - Face to Bore Undercut
11 - Clutch Pilot Bore
12 - Pilot Bore Chamfer
13 - Clutch Mounting Face
14 - Top Relief
15 - Clutch Mounting Hole
16 - Radial Clutch Pin Hole
17 - Ring Gear Trunnion
18 - Ring Gear
19 - Trunnion Chamfer
20 - Web Relief
21 - Supplier ID
22 - Crankshaft Mounting Hub
23 - Crankshaft Mounting Bore
24 - Crankshaft Mounting Undercut
25 - Crankshaft Mounting Face
26 - Crankshaft Mounting Hole Counterbore
27 - Crankshaft Mounting Hole Countersink
28 - Crankshaft Mounting Hole
29 - Puller Hole
30 - Puller Hole Counterbore
31 - Balance Holes
32 - Engine
33 - Flywheel Housing
34 - Clutch
35 - Starter Mounting

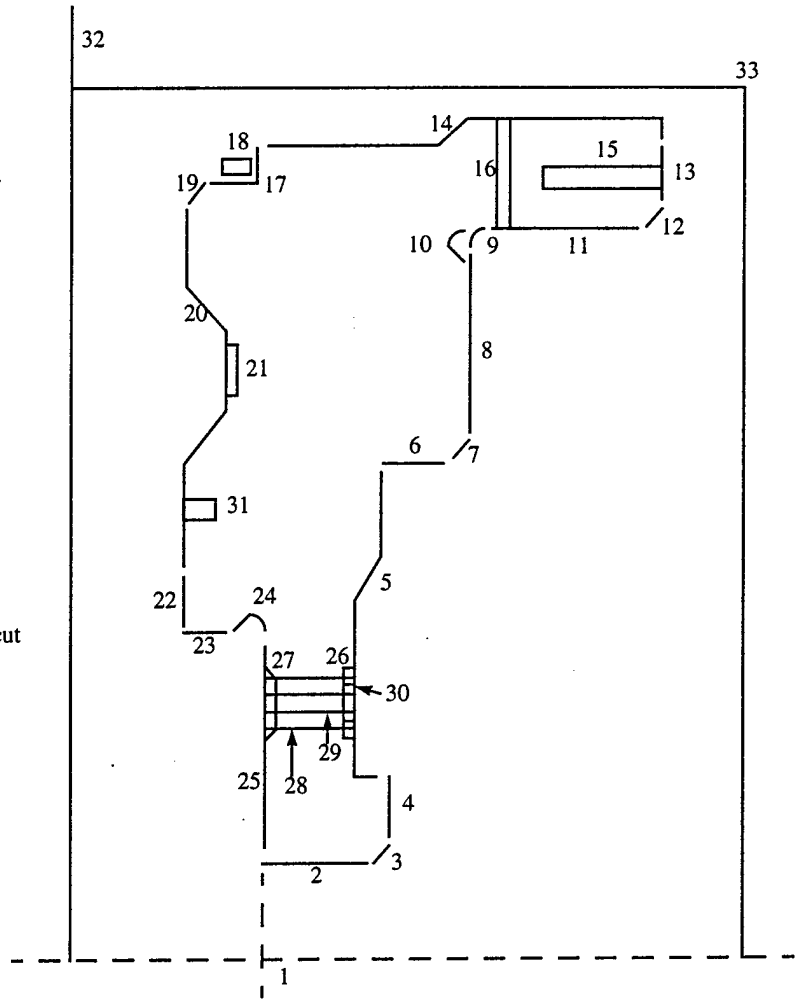


Figure 5-2. Flywheel Features

5.2.2 Flywheel Knowledge Sources

The knowledge sources implemented for the flywheel are based on the Cummins Engine Company Flywheel Design Standard which incorporates Society of Automotive Engineers (SAE) and International Organization for Standardization (ISO) design standards [CUMM94]. While the design standards enforce some strict design rules that cannot be violated, most provide recommended parameter values, feature interactions, and engineering analyses for flywheel design. The variety of engineering knowledge contained within the standards provided a broad basis for testing the knowledge source framework.

```
class CrnkMtgHubPos extends KnowledgeSource
{
    public String[] triggerConditions()
    {
        String[] trigger_conditions = { "Crankshaft Mounting Hub", "Crankshaft Mounting Bore"}
    }

    public void performAction()
    {
        Feature Bore, Hub;
        Bore = Model.getFeature("Crankshaft Mounting Bore");
        Hub = Model.getFeature("Cranksaft Mounting Hub");
        Hub.Attach(0,Bore,0);
    }
    public boolean canPerformAction()
    {
        Feature Hub;
        if (Model.featureIsThere("Crankshaft Mounting Hub"))
            if (Model.featureIsThere("Crankshaft Mounting Bore")){
                Hub = Model.getFeature("Crankshaft Mounting Hub");
                if (!Hub.positionRelationIsSet())
                    return true;
            }
        return false;
    }
}
```

Figure 5-3. Source Code for Hub Positioning Knowledge Source

Over 140 knowledge sources were developed for the flywheel domain, and they are categorized into the modification, advisory, and analysis types.

The modification knowledge sources interact with the model in three ways. First, positioning knowledge sources automatically define the positioning relations for the various features as they are added to the model. For example, referring again to Figure 5-2, when the crankshaft mounting hub is added to the model, a knowledge source automatically defines a positioning relation to attach it to the crankshaft mounting bore. These knowledge sources were chosen to present themselves automatically because the positions of the features are well established for the flywheel. The designer can, however,

modify the positioning relations during the design process to test unique positioning scenarios. The Java source code for this knowledge source is shown in Figure 5-3.

The second type of modification knowledge source implemented for the flywheel involves those rules that define the parameter values of particular features within the model. Several knowledge sources apply these standards to the model and present themselves to the designer interactively. For example, the SAE flywheel standards define the number of radial clutch pin holes based on the clutch type. The source code for this knowledge source is shown in Figure 5-4.

The third type of modification knowledge sources established for the flywheel involve those that add features to the model or remove them based on certain characteristics of the model. If, for example, the designer chooses to add a face-to-bore fillet to a flywheel that already has a face-to-bore undercut, the undercut must be removed from the model before the fillet is added. Several knowledge sources have been implemented to identify these types of situations and modify the model accordingly. Once again, these knowledge sources present themselves interactively, allowing the designer to control their application.

The second type of knowledge source, the advisory knowledge source, was utilized to present information to the designer for domain knowledge that did not specify specific changes to features or the model. For example, the Cummins Flywheel Design Standard recommends a range for the crankshaft mounting undercut dimensions based on the engine type. Ranges of dimension values, not specific values, are recommended, so the knowledge source only presents those ranges to the designer and leaves the responsibility of modifying the parameter values to the designer.

The final type of modification knowledge source implemented for the flywheel design process is the analysis knowledge source. One example of this type of knowledge source involves the diameter of the crankshaft mounting hole countersink and the remaining clearance around the holes, as shown in Figure 5-5. The surface area around the holes

```
class PinHoleNumber extends KnowledgeSource implements Dialogable
{
    public String[] triggerConditions()
    {
        String[] trigger_conditions = { "Clutch", "Radial Clutch Pin Holes"}
    }
    public void createDialog(DesignEnv FBDE)
    {
        KSD = new YesNoDialog(FBDE,this);
    }
    public void initDialog(KnowledgeSourceDialog KSD)
    {
        KSD.setText("The number of pin holes should be 12 for this clutch type. Do you wish to
          change it?");
    }
    public void performAction()
    {
        Feature PinHoles= Model.getFeature("Radial Clutch Pin Holes");
        PinHoles.setParameterValue("Number of Holes", 12);
    }
    public boolean canPerformAction()
    {
        Feature Clutch, Holes;
        if (Model.featureIsThere("Clutch"))
            if (Model.featureIsThere("Radial Clutch Pin Holes")){
                Hub = Model.getFeature("Crankshaft Mounting Hub");
                Clutch = Model.getFeature("Clutch");
                if (Clutch.getParameterValue("Type") == Clutch.POT){
                    if (Clutch.getParameterValue("Number of Holes") != 12)
                        return true;
                    }
            }
        return false;
    }
}
```

Figure 5-4. Java Source Code for Parameter Modification Knowledge Source

must be a certain percentage of the hole diameter. The knowledge source responsible for applying this rule calculates whether the necessary clearance is available based on the positions and dimensions of the crankshaft mounting undercut, the crankshaft mounting bore, and the crankshaft mounting holes. If the necessary clearance is not available, the knowledge source calculates the possible diameter changes of the holes and the bore and

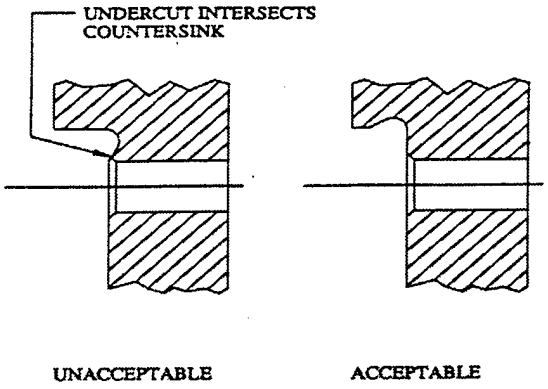


Figure 5-5. Crankshaft Mounting Hole Clearance (Courtesy of Cummins Engine Company, Inc.)

presents them to the designer. Just as for the advisory knowledge sources, the designer is then responsible for making the appropriate changes to the model.

5.3 The Flywheel Design Process

The design process for the flywheel begins when the designer instantiates a new model. The system prompts the designer to specify the domain of the model to create. As shown in Figure 5- 6, after the designer has specified the design domain, in this case the flywheel, the system dynamically loads both the flywheel knowledge sources and feature

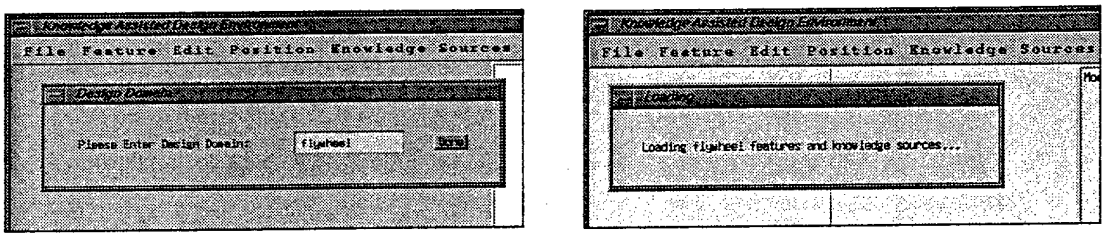


Figure 5-6. Domain Specification and Dynamic Knowledge Source and Feature Loading

set and establishes a design assistant to begin monitoring any modifications to the model. The design process proceeds from the start-up feature modification stage to feature modification.

5.3.1 Start-Up Feature Modification

After the FBDE and the design assistant load the flywheel features and knowledge sources into the system, respectively, the FBDE prompts the designer, as shown in Figure 5-7, to define the parameter values for the various start-up features. Four features have been identified as the start-up features for the flywheel: the engine family, the clutch, the flywheel housing, and the starter mounting. At this time, the designer can either choose to

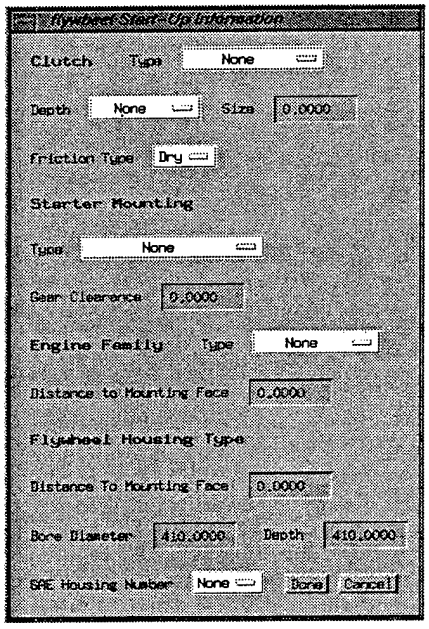


Figure 5-7. Start-Up Feature Modification

define these features or cancel to proceed directly to the functional and geometric feature modification stages of the design process. If the designer chooses to define the parameter values, the features need not be completely defined. For this example, the parameter

values are partially defined. Once the designer is satisfied with the parameter values, he or she disposes the dialog and the design process enters the feature modification stage.

5.3.2 Feature Modification

The FBDE adds the start-up features to the model, once they have been defined. The design assistant observes these modifications and presents them to the knowledge sources. Several knowledge sources are triggered by the modifications and found to be applicable to the current state of the model. Consider the knowledge source shown in Figure 5-8. The

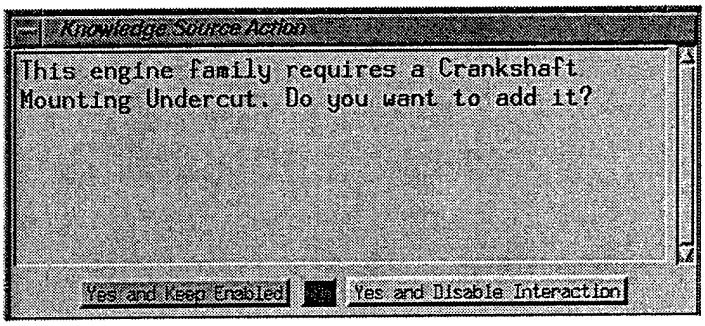


Figure 5-8. Knowledge Source Presentation Dialog

design assistant presents the designer with a knowledge source that recommends that a crankshaft mounting undercut should be added to the model because of the engine family chosen during the start-up feature modification. The designer has three options for applying this knowledge source to the model. First, the designer can accept the knowledge source's action and keep its interactive presentation enabled. The knowledge source will continue to contribute to the design process and present itself interactively. For example, if the designer were to delete the undercut, this knowledge source would once again present itself. Second, the designer can decline to accept the knowledge source's action. If this option is selected, the system prompts the designer, as shown in Figure 5-9, to provide an

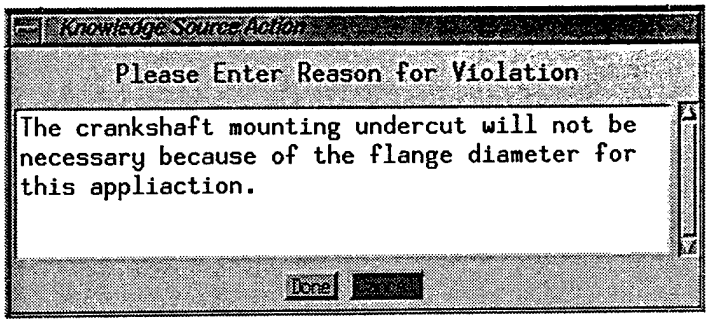


Figure 5-9. Rule Violation Explanation Dialog

explanation of why the knowledge source's actions were not accepted. The system stores this information in a design log along with the model data that can be accessed at a later time. If this option is selected, the system disables the knowledge source, and it will not contribute to the design process until the designer chooses to enable it. Third, the designer can accept the action and disable the knowledge source from interactively presenting itself during the design process. The knowledge source participates in the design process but applies itself automatically.

After all of the start-up features have been added to the model and the applicable knowledge sources have presented themselves to the designer, the designer is free to add, remove, and edit features in any arbitrary order. The design assistant reacts to any of these design changes and presents the applicable knowledge sources to the designer. If, for example, the designer chooses to add the crankshaft mounting bore to the model, three events would occur. First, the FBDE adds the bore to the model structure. Second, the design assistance recognizes this change and presents it to the knowledge sources. The design assistant then presents the applicable knowledge sources to the designer, and the knowledge sources make their respective feature modifications. In this case, some of the bore's parameter values are changed, and both the bore and the undercut are positioned in the model. The third event is the graphical presentation of those features that are

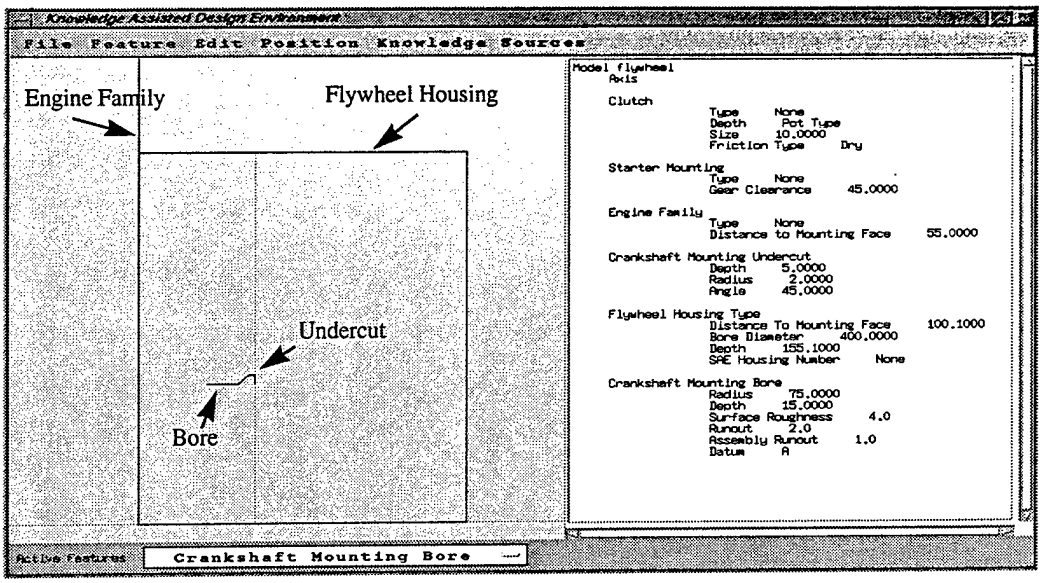
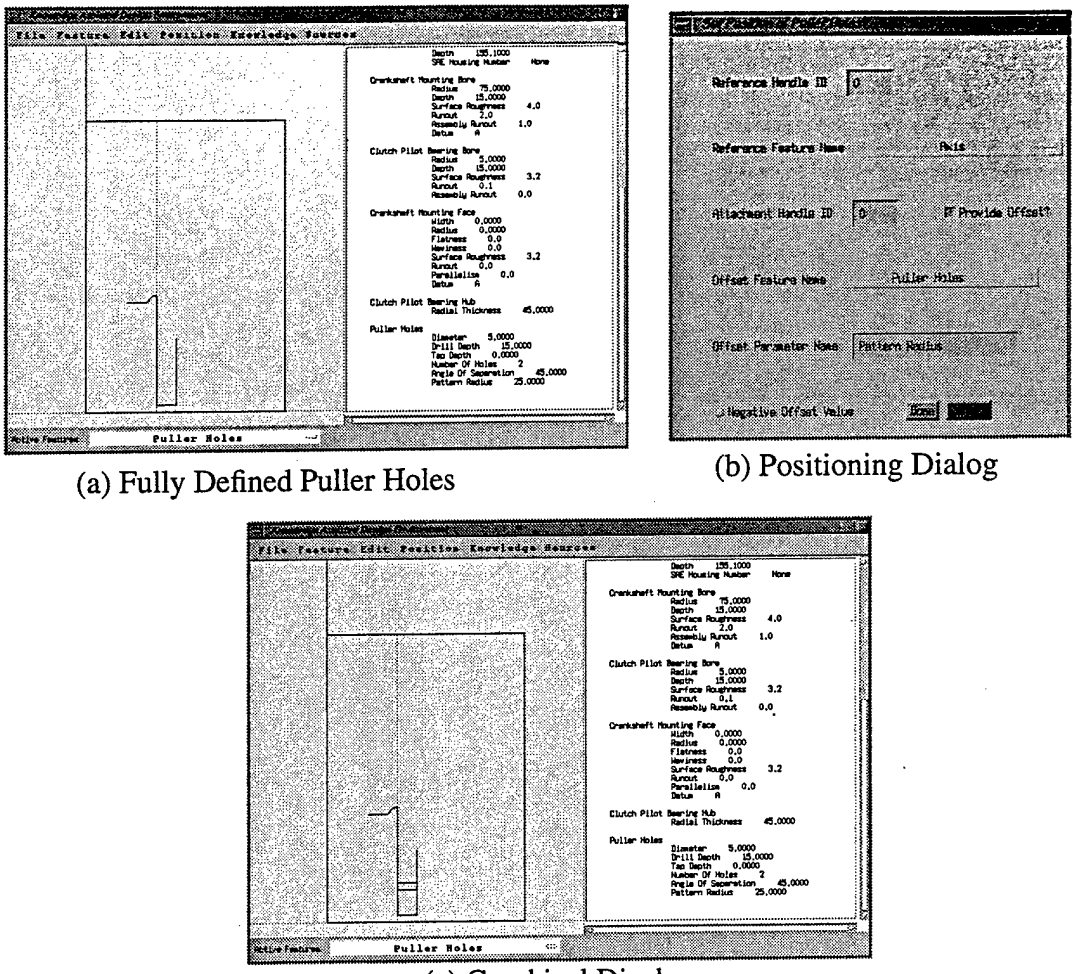


Figure 5-10. Graphical Presentation of Features

completely defined and positioned within the model. As shown in Figure 5-10, the designer has completely defined the engine family, flywheel housing, crankshaft mounting bore, and crankshaft mounting undercut. They are completely positioned in the model and are presented graphically. It should also be noted that the other features that have been added to the model are presented textually.

Most of the flywheel feature positioning relations are established by automatic knowledge sources. If the knowledge sources are disabled, however, or the positioning relations are not completely defined by the knowledge sources, the designer can define the positioning relations for a particular feature. Consider the puller hole feature that the designer has added to the model and defined completely, as shown in Figure 5-11(a). The feature is not presented graphically because it has not been positioned. The designer can position the feature horizontally and vertically using the positioning dialog shown in Figure 5-11(b). In this example, the designer is positioning the vertical position of the puller holes with respect to the axis and offset from the axis by the puller holes' pattern



(a) Fully Defined Puller Holes

(b) Positioning Dialog

(c) Graphical Display

Figure 5-11. Puller Hole Position Modification

radius. The designer defines the puller hole horizontal position similarly, and the system displays the holes as shown in Figure 5-11(c).

At any time during the design process, the designer can select a feature in the model and edit its parameter values. In doing so, the system presents a dialog to the designer in which the various parameters of the feature can be modified, as shown for editing the clutch in Figure 5-12. Parameters with continuous values are presented with their value field as an editable text area, such as the clutch size. The discrete values, such as the clutch

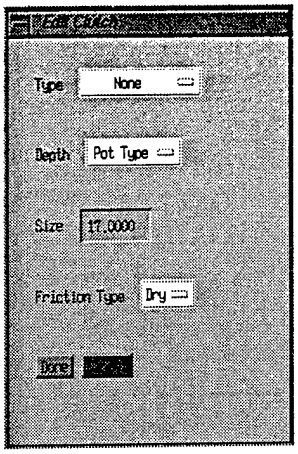


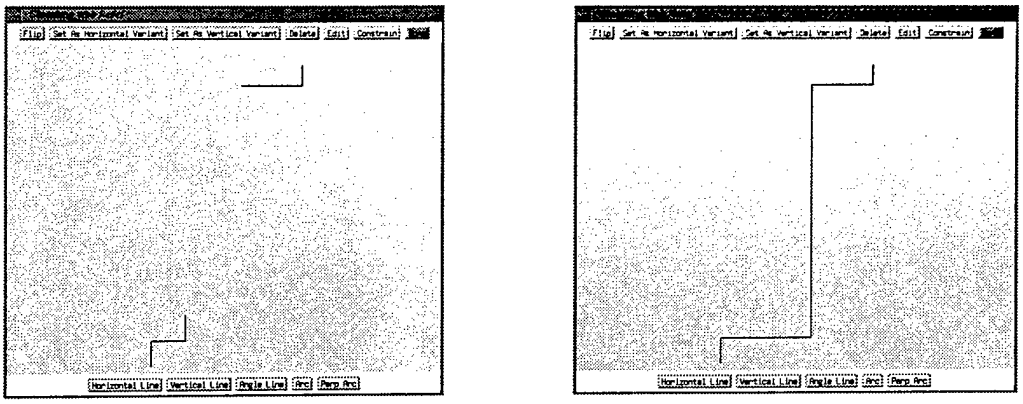
Figure 5-12. Clutch Edit Dialog

friction type, are presented to the designer with the possible parameter values from which to choose.

5.3.3 Variable Geometry Feature Modification

If the designer selects a variable geometry feature to edit, he or she can edit or define its geometric elements along with its parameter values. If the feature's position is dependent, the designer cannot define its geometry until both of the dependency declarations have been assigned. Consider the flywheel's web relief, which is a dependent position, variable geometry feature. Once the designer has added the ring gear trunnion and the crankshaft mounting hub to and positioned them in the model, the web relief's geometry can then be defined.

The designer defines the feature's geometry in the design environment shown in Figure 5-13. As the geometric elements are added to the model, the designer can edit their parameter values and define the horizontal and vertical variants. In this case, the horizontal line has been assigned as the horizontal variant, and the vertical line has been assigned as the vertical variant. The designer can then constrain the feature and view the resulting

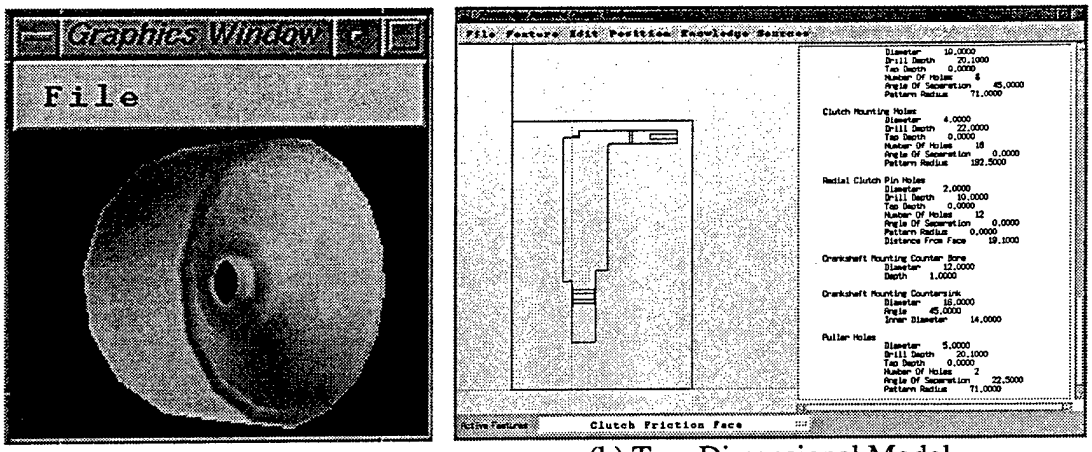


(a) Geometry Specification (b) Constrained Geometry

Figure 5-13. Web Relief Geometry Specification

geometry, as shown in Figure 5-13(b). Once the geometry has been completely defined, then designer exits this design environment and returns to the feature-based design system.Creating A Solid Model

After the designer has added all of the necessary features to the flywheel model, the system can build a three dimensional solid model of the flywheel to better visualize the completed design. The prototype design system creates the solid model using the Java-TWIN solid modeling package and renders it for presentation to the designer using a Java port of Silicon Graphics' Open-GL graphics library [CADL91]. The system builds the flywheel solid model by rotating the cross section outline 360 degrees about the flywheel's axis. The rendering of the completed three-dimensional flywheel model is shown along with the two-dimensional cross section and textual representations in Figure 5-14.



(a) Three-Dimensional Model (b) Two-Dimensional Model

Figure 5-14. Three-Dimensional, Two-Dimensional, and Textual Model Visualization

CHAPTER 6

CONCLUSION

Computer-aided design systems have become an integral part of the modern design process. While today's CAD tools provide support for drafting, geometric modeling, and computational engineering analysis, they do not serve as interactive, knowledgeable support tools for the design process. This can be attributed to fact that conventional CAD methodologies do not integrate experiential engineering knowledge into the design process. Designers must draw upon their past experiences and design knowledge during the design process, just as they did before the advent of CAD.

This research developed a computer-aided design methodology for integrating engineering knowledge in a CAD environment that provides dynamic, interactive, knowledgeable design assistance for a design process that is completely directed by the designer. The contributions of the knowledge assisted design methodology can be summarized in two areas. First, the model and knowledge representations have resolved many of the inadequacies of the representations used in past AI-base CAD systems. They are extensible to varied design domains by the design engineer. The representations also facilitate the desired design assistance during the design process. In addition, the product model created during the design process defines both the geometric and non-geometric properties of the design artifact, unlike past research efforts in this field. The second area of contribution is the development of a domain independent framework for building a knowledge assisted design environment. The resulting system integrates the designer, the model, and the engineering knowledge in an environment in which the designer maintains

control of a design process interactively supported by the engineering knowledge. The knowledge sources that apply themselves opportunistically to the developing design solution. They also serve as a repository for designers' knowledge well after they leave the organization.

6.1 Future Research and Development

While the knowledge assisted design methodology contributes to the research in this area, there are a number of areas in which further research and development could improve the methodology. The methodology presents a sound, domain independent framework that could be applied to any design domain, provided the appropriate set of features and knowledge sources were implemented for that domain. It was, however, only validated with the engine flywheel design example. The flywheel proved to be a broad and well documented test scenario. A large amount of engineering information was available for the flywheel product model, and the variability in the flywheel features and creativity required during the design process fit well into a design assistance framework. The flywheel features and design rules were well documented prior to the start of the research, and the axisymmetric, two dimensional nature of the flywheel's geometry also greatly reduced the geometric complexity of the positioning algorithms and feature definitions. To further develop the methodology and reveal any development issues not encountered during the flywheel implementation, the methodology should be tested using a variety of design domains.

In particular, two specific properties of design domains should be tested. First, a domain in which the geometry cannot be simplified to a two-dimensional cross-section representation would test the implications of more complex geometric properties of the model with the knowledge source interactions. This would also necessitate the implementation of a constraint management package within the design environment,

thereby increasing the capabilities of the designer and the knowledge sources to define relationships between features within the model. The second domain property that should be validated is a feature definition process that is not as clear as it was for the flywheel. Although the features developed for this research proved successful for the flywheel, this modeling paradigm may not fit well into other design domains. In that case, an alternate model structure would have to be developed to accommodate those situations.

Another important area of future research that was somewhat addressed in this work is the concept of using distributed knowledge sources and feature bases from a centrally located knowledge assisted design environment, or using centrally located knowledge and feature bases from distributed design environments. While this was not implemented in the prototype design environment, the framework was specifically designed to facilitate the development of these capabilities. The Java programming language was used not only because of its platform independent properties and well developed object-oriented programming framework and because of the sophisticated networking capabilities built into the language, such as Remote Method Invocation (RMI). The knowledge sources and features are both maintained as separate objects within the design environment framework, and extending the system to utilize these objects if they are located on remote computer systems would not be a difficult task. However, the use of distributed objects does bring about a number of other research issues such as access and control that would need to be addressed.

While the use of distributed objects in the knowledge assisted design environment may bring about some control issues not yet encountered, another area of research for the design methodology in its current, non-distributed state is the issue of knowledge source action control. Although the control algorithm can identify loops and conflicts during the knowledge source application process, more sophisticated control algorithms could be developed that resolve some of the obvious conflicts without involving the direct action of

the designer. In addition, if the number of knowledge sources increases by a factor of 10 or 100, more sophisticated algorithms would have to be developed to increase the efficiency of the knowledge source query process.

A research issue that would greatly contribute to the capabilities of the methodology is the development of knowledge source and feature development environments. One of the primary issues in this work was to develop model and knowledge representations that could be adapted to any design domain by an engineer without involving a specialized knowledge engineer. While the representations are relatively straight forward and easy to extend, the creation of knowledge source and feature development environments would further reduce the complexity involved in extending the knowledge assisted design methodology to a specific domain. In particular, a knowledge source development environment could verify that a knowledge source's action does not conflict with any other knowledge source action before adding it to the set of available knowledge sources.

In addition to the suggestions for future research presented in this section, a number of conventional CAD capabilities such as finite element analysis and manufacturability analysis could be added to the design environment to further support the entire design process. The intent of the knowledge assisted design methodology is to provide a flexible framework that can be applied to numerous design scenarios in which the integration of experiential engineering knowledge and conventional CAD methodologies will contribute to the overall success of a completed product model.

LIST OF REFERENCES

LIST OF REFERENCES

[ABU94] Abu-Hanna, A., Jansweijer, W., Benjamins, R. and Wielinga, B., "Functional Models in Perspective: Their Characteristics and Integration in Multiple Model-Based Diagnosis," *Applied Artificial Intelligence*, Vol. 8, pp. 219-237, 1994.

[AKMA94] Akman, V., ten Hagen, P. J. W. and Tomiyama, T., "Desirable Functionalities of Intelligent CAD Systems," *Intelligent Systems in Design and Manufacturing*, ASME Press, New York, 1 ed., pp. 117-138, 1994.

[ANDE86] Anderson, D. C., "Closing the Gap: A Workstation-Mainframe Connection," *Computers in Mechanical Engineering*, Vol. 4, No. 6, pp. 16-24, 1986.

BARD93] Bardasz, T. and Zeid, I., "DEJAVU: Case-Based Reasoning for Mechanical Design," *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, Vol. 7, No. 2, pp. 111-124, 1993.

[BATA93] Batanov, D. N. and Lekova, A. K., "Data and Knowledge Integration Through the Feature-Based Approach," *Artificial Intelligence in Engineering*, Vol. 8, pp. 77-83, 1993.

[BAYL95] Bayliss, D. C., Akueson, R. and Knight, J. A. G., "Concurrent Engineering Philosophy Implemented Using Computer Optimized Design," *Journal of Engineering Manufacture*, Vol. 209, No. B3, pp. 193-199, 1995.

[BIJL87] Bijl, A., "Strategies for CAD," *Intelligent CAD Systems I*, Springer-Verlag, Berlin, pp. 2-19, 1987.

[BRIN95] Brinkop, A., Laudwein, N. and Maasen, R., "Routine Design for Mechanical Engineering," *AI Magazine*, Vol. 16, No. 1, pp. 74-85, 1995.

[BROW92] Brown, D. and Chandrasekaran, B., "Knowledge and Control for the Mechanical Design of an Expert System," *IEEE Computer*, Vol. 19, No. 7, pp. 92-100, 1992.

[BROW95] Brown, K. N., Williams, J. H. and McMahon, C. A., "Conceptual Geometric Reasoning by the Manipulation of Models Based on Prototypes," *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, Vol. 9, No. 5, pp. 367-385, 1995.

[BURC87] Burchard, R. L., *Feature-Based Geometric Constraints Applied to Constructive Solid Geometry*, MS Thesis, Purdue University, 1987.

[BUSH87] Bushnell, M. L. and Director, S. W., "ULYSSES - A Knowledge-Based VLSI Design Environment," *Artificial Intelligence in Engineering*, Vol. 2, No. 1, pp. 33-41, 1987.

[CADL91] *Twin Solid Modeling Package Reference Manual*, CADLAB, School of Mechanical Engineering, Engineering Research Center, Purdue University, September 1991.

[CANT95] Cantzler, O., Mekhilef, M. and Bocquet, J.-C., "A Systemic Approach to Corporate Knowledge: An Ontology to Process Modeling in a Design Department," Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, Vol. 1, pp. 153-158, 1995.

[CHAM95] Chambers, T. L. and Parkinson, A. R., "Knowledge Representation and Conversion for Hybrid Expert Systems," Proc. ASME Design Engineering Technical Conferences, 21st Annual Design Automation Conference, American Society of Mechanical Engineers, Boston, Massachusetts, DE (Series), Vol. 82, pp. 9-16, 1995.

[CHEN95] Chen, Y., Miller, A. and Sevenler, K., "Knowledge-Based Manufacturability Assessment: An Object-Oriented Approach," *Journal of Intelligent Manufacturing*, Vol. 6, No. 5, pp. 321-337, 1995.

[CHIT94] Chittaro, L., Tasso, C. and Toppano, E., "Putting Functional Knowledge on Firmer Ground," *Applied Artificial Intelligence*, Vol. 8, No. 2, pp. 239-258, 1994.

[CORB86] Corby, O., "Blackboard Architectures in Computer Aided Engineering," *Artificial Intelligence in Engineering*, Vol. 1, No. 2, pp. 95-98, 1986.

[CUMM94] *Flywheel Product Technology Practice (Design)*, Cummins Engineering Standard 98016, Cummins Engine Company, Inc., Columbus, IN, December 1994.

[DELO95] Deloule, F. and Roche, C., "Ontologies & Knowledge Representation," Proc. 1995 IEEE International Conference on Systems, Man and Cybernetics, IEEE, Vancouver, British Columbia, Vol. 5, pp. 3857-3862, 1995.

[DEMA92] Demain, A. and Zucker, J., "Prototype-Oriented Representation Of Engineering Design Knowledge," *Artificial Intelligence in Engineering*, Vol. 7, pp. 47-61, 1992.

[DIAZ94] Diaz-Claderon, A., Fenves, S., et al., "Computer-Based Advisors for Environmentally Conscious, 'Green' Product Design," *Computing in Civil Engineering*, Vol. 2, pp. 1497-1504, 1994.

[DIXO95] Dixon, J. R., "Knowledge-Based Systems for Design," *Journal of Mechanical Design*, Vol. 117B, No. 6, pp. 11-16, 1995.

[DIXO87] Dixon, J. R., Cunningham, J. J. and Simmons, M. K., "Research in Designing with Features," Proc. IFIP TC 5/WG 5.2 Workshop on Intelligent CAD, Elsevier Science Publishing Company, Boston, MA, Vol. 1, pp. 137-148, 1987.

[DIXO84] Dixon, J. R., Simmons, M. K. and Cohen, P. R., "An Architecture for Application of Artificial Intelligence to Design," Proc. ACM/IEEE 21st Design Automation Conference, Albuquerque, NM, pp. 634-640, 1984.

[DOME93] Domeshek, E. and Kolodner, J., "Using the Points of Large Cases," *Artificial Intelligence for Engineering Design and Manufacture*, Vol. 7, No. 2, pp. 87-96, 1993.

[DOWL94] Dowlatshahi, S., "A Comparison of Approaches to Concurrent Engineering," The International Journal of Advanced Manufacturing Technology, Vol. 9, No. 2, pp. 106-113, 1994.

[DUFF96a] Duffy, A. H. B. and Duffy, S. M., "Learning for Design Reuse," Artificial Intelligence for Engineering Design, Analysis and Manufacturing, Vol. 10, No. 2, pp. 139-142, 1996.

[DUFF96b] Duffy, A. H. B. and Duffy, S. M., "Sharing the Learning Activity Using Intelligent CAD," *Artificial Intelligence for Engineering Design and Manufacture*, Vol. 10, No. 2, pp. 83-100, 1996.

[DYM88] Dym, C. L., Henchey, R. P., Delis, E. A. and Gonick, S., "Representation and Control Issues in Automated Architectural Code Checking," *Computer-Aided Design*, Vol. 20, No. 3, pp. 137-145, 1988.

[ENGE88] Engelmore, R. and Morgan, T., *Blackboard Systems*, 1 ed., Addison-Wesley, New York, 1988.

[FALT96] Faltings, B. and Sun, K., "FAMING: Supporting Innovative Mechanism Shape Design," *Computer-Aided Design*, Vol. 28, No. 3, pp. 207-216, 1996.

[GALL95] Galle, P., "Towards Integrated, 'Intelligent,' and Compliant Computer Modeling of Buildings," *Automation in Construction*, Vol. 4, No. 3, pp. 189-211, 1995.

[GERO88] Gero, J. S., Maher, M. L. and Zhang, W., "Chunking Structural Design Knowledge as Prototypes," Artificial Intelligence in Engineering: Design, Computational Mechanics Publications, Southampton, pp. 3-21, 1988.

[GERO94] Gero, J. S. and Rosenman, M. A., "The What, the How, and the Why in Design," *Applied Artificial Intelligence*, Vol. 8, pp. 199-218, 1994.

[GOEL89] Goel, A. K. and Chandraskaran, B., "Use of Device Models in Adaptation of Design Cases," Proc. DARPA Workshops on Case-Based Reasoning, Morgan Kaufmann, San Mateo, CA, pp. 100-109, 1989.

[GOEL96] Goel, A. K. and Stroulia, E., "Functional Device Models and Model-Based Diagnosis in Adaptive Design," *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, Vol. 10, No. 4, pp. 355-370, 1996.

[GRUB93] Gruber, T. R., Toward Principles for the Design of Ontologies Used for Knowledge Sharing, Stanford Knowledge Systems Laboratory, Technical Report, KSL 93-04, 1993.

[HAN95] Han, C., Douglas, J. M. and Stephanopoulos, G., "Agent-Based Approach to a Design Support System for the Synthesis of Continuous Chemical Processes," *Computers and Chemical Engineering*, Vol. 19, Supplement, pp. S63-S69, 1995.

[HAYE92] Hayes-Roth, B., Washington, R., et al., "Guardian. A Prototype Intelligent Agent for Intensive-Care Monitoring," *Artificial Intelligence in Medicine*, Vol. 4, No. 2, pp. 165-185, 1992.

[HINK95] Hinkle, D. and Toomey, C., "Applying Case-Based Reasoning to Manufacturing," *AI Magazine*, Vol. 16, No. 1, pp. 65-73, 1995.

[HUA96] Hua, K., Faltings, B. and Smith, I., "CADRE: Case-Based Geometric Design," *Artificial Intelligence in Engineering*, Vol. 10, No. 2, pp. 171-183, 1996.

[JAGA89] Jagannathan, V., Dodhiawala, R. and Baum, L., Ed., *Blackboard Architectures and Applications, Perspectives in Artificial Intelligence*, Academic Press, Inc., San Diego, CA, 1989.

[JOSK96] Joskowicz, L. and Neville, D., "A Representation Language for Mechanical Behavior," *Artificial Intelligence in Engineering*, Vol. 10, No. 2, pp. 109-116, 1996.

[KIMU95] Kimura, F. and Suzuki, H., "Representing Background Information for Product Description to Support Product Development Process," *Annals of the CIRP*, Vol. 44, No. 1, pp. 113-116, 1995.

[KUMA95] Kumar, H. S. and Krishnamoorthy, C. S., "A Framework for Case-Based Reasoning in Engineering Design," *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, Vol. 9, No. 3, pp. 161-182, 1995.

[KUSI95] Kusiak, A. and N., L., "Decomposition and Representation Methods in Mechanical Design," *Journal of Mechanical Design*, Vol. 117B, June, pp. 17-24, 1995.

[MACC90] MacCallum, K. J., "Does Intelligent CAD Exist?," *Artificial Intelligence in Engineering*, Vol. 5, No. 2, pp. 55-64, 1990.

[MAHE95] Maher, M. L., Balachandran, M. B. and Zhang, D. M., *Case-Based Reasoning in Design*, 1 ed., Lawrence Erlbaum Associates, Inc., Publishers, New Jersey, 1995.

[MARC88] Marcus, S., Stout, J. and McDermott, J., "VT: An Expert Elevator Designer That Uses Knowledge-Based Backtracking," *AI Magazine*, Vol. 9, No. 1, pp. 95-112, 1988.

[MASO95] Masood, S. H. and Lim, B. S., "Concurrent Intelligent Rapid Prototyping Environment," *Journal of Intelligent Manufacturing*, Vol. 6, No. 5, pp. 291-310, 1995.

[MAYE88] Mayer, A. K. and Lu, S. C.-Y., "An AI-Based Approach for the Integration of Multiple Source of Knowledge to Aid Engineering Design," *ASME Journal of Mechanisms, Transmissions, and Automation in Design*, Vol. 110, No. 3, pp. 316-323, 1988.

[MESS94] Messimer, S. L. and Henshaw, J., "Composites Design and Manufacturing Assistant," *Int. J. of Materials and Product Technology*, Vol. 9, No. 1/2/3, pp. 105-115, 1994.

[MITT86] Mittal, S., Dym, C. L. and Morjaria, M., "PRIDE: An Expert System for the Design of Paper Handling Systems," *IEEE Computer*, Vol. 19, No. 7, pp. 102-114, 1986.

[OXMA93] Oxman, R. E. and M., O. R., "Remembrance of Things Past: Design Precedents in Libraries," *Automation in Construction*, Vol. 2, pp. 21-29, 1993.

[PACK94] Packer, S. M. and Epstein, R. A., "Knowledge Based Engineering of Mold Transport Substructure at Sikorsky Aircraft," *Annual Forum Proceedings - American Helicopter Society*, Vol. 2, No. 1, pp. 709-718, 1994.

[RIIT88] Riitahuhta, A., "Systematic Engineering Design and Use of an Expert System in Boiler Plant Design," Proc. ICED International Conference on Engineering Design, Budapest, Hungary, pp. 95-110, 1988.

[RODE93] Roderman, S. and Tsatsoullis, C., "PANDA: A Case-Based System to Aid Novice Designers," *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, Vol. 7, No. 2, pp. 125-133, 1993.

[RODR94] Rodriguez, J. and Hart, P., "Preliminary Shape Definition of Structural Components Using a Prototype Knowledge-Based Expert System," *Engineering with Computers*, Vol. 11, No. 2, pp. 103-113, 1994.

[ROGE95] Rogers, K. J., Priest, J. W. and Haddock, G., "The Use of Semantic Networks to Support Concurrent Engineering in Semiconductor Product Development," *Journal of Intelligent Manufacturing*, Vol. 6, No. 5, pp. 311-319, 1995.

[ROSE94] Rosenman, M. A., Gero, J. S. and Maher, M. L., "Knowledge-Based Design Research at the Key Centre of Design Computing," *Automation in Construction*, Vol. 3, No. 2, pp. 229-237, 1994.

[ROY95] Roy, U., Balaji, B., Sarathy, S. and Graham, P., "Development of an Intelligent Product Design System: Integration Strategies," *Applied Artificial Intelligence*, Vol. 9, No. 6, pp. 563-585, 1995.

[SCHE93] Scherer, R. J. and Katranuschkov, P., "Architecture of an Object-Oriented Product Model Prototype for Integrated Building Design," Proc. of the 5th International Conference on Computing in Civil and Building Engineering, ASCE, Anaheim, CA, Vol. ICCCBE, pp. 393-400, 1993.

[SHAH95] Shah, J. J. and Mantyla, M., *Parametric and Feature-Based CAD/CAM: Concepts, Techniques, Applications*, 1 ed., John Wiley & Sons, New York, 1995.

[SOBO91] Sobolewski, M., *Object-Oriented Knowledge Bases in Engineering Applications*, Concurrent Engineering Research Center, West Virginia University, Technical Report Research Note, CERC-TR-RN-91-013, 1991.

[SRIA86] Sriram, D., "DESTINY: A Model for Integrated Structural Design," *Artificial Intelligence in Design*, Vol. 1, No. 2, pp. 109-116, 1986.

[STEI92] Steinberg, L. I., "Design as Top-Down Refinement Plus Constrain Propogation," *Artificial Intelligence in Engineering Design*, Academic Press, Inc., San Diego, CA, pp. 251-272, 1992.

[SYCA92] Sycara, K. and Navinchandra, D., "Retrieval Strategies in a Case-Based Design System," *Artificial Intelligence in Engineering Design*, Academic Press, San Diego, pp. 145-163, 1992.

[THOM95] Thomas, C. and Rozenblit, J. W., "Projection-Based Knowledge Representation for Concurrent Engineering," Proc. 1995 IEEE International Conference on Systems, Man and Cybernetics, IEEE, Vancouver, British Columbia, Vol. 5, pp. 3863-3868, 1995.

[VENK86] Venkatasubramanian, V. and Chen, C. F., "A Blackboard Architecture for Plastics Design," *Artificial Intelligence in Engineering*, Vol. 1, No. 2, pp. 117-122, 1986.

[YAGI91] Yagiu, T., *Modeling Design Objects and Processes*, 1 ed., Springer-Verlag, Berlin, 1991.

[YEH96] Yeh, S., Kamran, M., Terry, J. and Nnaji, B. O., "A Design Advisor for Sheet Metal Fabrication," *IIE Transactions*, Vol. 28, No. 1, pp. 1-10, 1996.